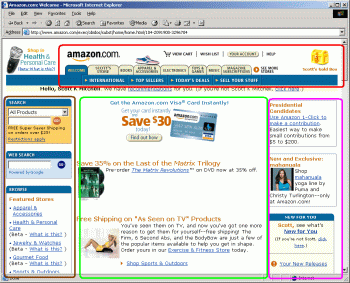
**A Sneak Peak at MasterPages in ASP.NET 2.0**   
*By* [*Scott Mitchell*](http://www.4guysfromrolla.com/ScottMitchell.shtml)

**Introduction**  
When graphic designers create a layout for a Web site, they typically break down page layout into distinct regions. For example, as the screenshot to the right shows, all of Amazon.com's Web pages are divided into various sections, such as a header (circled in red), a left navigational region (circled in brown), a center content area (circled in green), and a right "recent news" area (circled in purple).

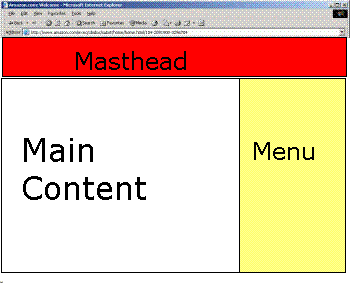
When building an ASP.NET 1.x Web application, you'll usually use a "header" and "footer" User Control. The header User Control would contain the starting <html> element, along with content that appeared at the top of every page. The footer User Control would contain the shared content that appeared at the bottom of every page, like a site map, and closing </html> and related tags. This works well if your site layout can be easily divided into a top HTML portion and a bottom HTML portion, but what if this is not the case?

With ASP.NET 1.x, the options for creating a site with a consistent look and feel are substandard. Yes, you can use User Controls as described above, but you must manually add the standard User Controls to each page. Furthermore, User Controls not rendered in Visual Studio .NET as little gray boxes, thereby removing the WYSIWYG benefits of the VS.NET Designer. A better technique for providing a consistent site layout in ASP.NET 1.x is to use *MasterPages*. As discussed earlier in [Creating Template-Driven Web Sites with MasterPages](http://aspnet.4guysfromrolla.com/articles/040704-1.aspx), MasterPages provide a means to define a common look and feel with editable content regions. A page, then, can utilize a MasterPage, thereby inheriting the standardized look and feel, and leaving the page developer to customize the editable regions.

Unfortunately with ASP.NET 1.x there is no support for MasterPages in Visual Studio .NET. Thankfully this has been changed with the upcoming ASP.NET 2.0. In Visual Studio 2005, there is rich, WYSIWYG support for MasterPages. In this article we'll look at MasterPage support in Visual Studio 2005 and ASP.NET 2.0. This article does not delve into the basic premise of MasterPages - for that, refer to [Creating Template-Driven Web Sites with MasterPages](http://aspnet.4guysfromrolla.com/articles/040704-1.aspx). Additionally, this article was written during the Beta 1 time frame of ASP.NET 2.0; the screenshots/instructions discussed may or may not make it to the final version of ASP.NET 2.0/Visual Studio 2005. You can download the latest 2.0 beta bits from <http://labs.msdn.microsoft.com/>.

**A Quick MasterPages Overview**  
The purpose of MasterPages is to provide an easy way to create a site template. MasterPages work by designing an HTML page with editable regions. These editable regions, called *content placeholders* represent the portions of the template that will be customized on each page. The area outside of the content placeholders is common to all pages on the site and is not editable on a page-by-page basis.

For example, imagine we wanted to create a site template similar to the one shown in the screenshot below.

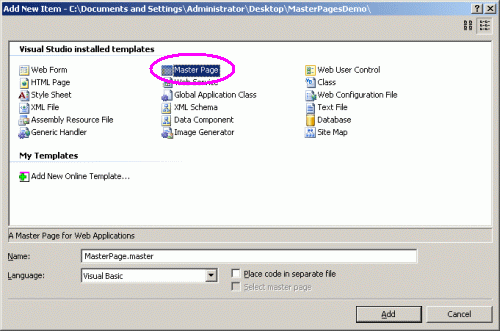


With this layout, let's assume that the masthead is static, displaying the same message on all pages. However, the main content and menu regions will differ on a page-by-page basis. Keep in mind that with ASP.NET 1.x and Visual Studio .NET our only options for creating such a consistent layout would be to embark on one of the following options:

* Implement the needed HTML on each and every page on our site
  + **Pros:** Visual Studio .NET will provide a rich, WYSIWYG design-time experience for each page.
  + **Cons:** If we want to change the look and feel of the site, we need to edit *every single page*!
* Implement the layout using a "top" and "bottom" User Control, which will be added to each page
  + **Pros:** Altering the site's layout only involves changing at most two files - the top and bottom User Controls.
  + **Cons:** Have to manually add the top and bottom User Controls to all pages; no WYSIWYG support.
* Utilize MasterPages support in ASP.NET 1.x
  + **Pros:** Can easily change the site layout by editing just the MasterPage; allows for a more fluid layout than kludging together top and bottom User Controls.
  + **Cons:** No WYSIWYG design-time support in VS.NET.

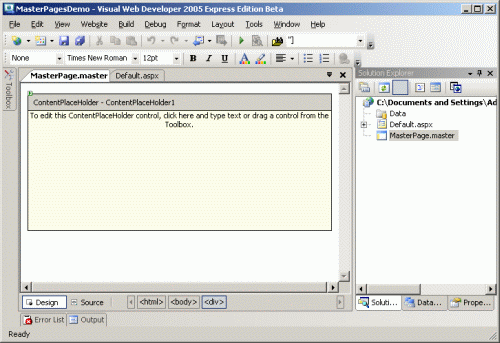
Fortunately with ASP.NET 2.0 and Visual Studio 2005, MasterPages offer rich design-time support. Therefore we can experience the pros of having a rich WYSIWYG design-time experience while keeping the site layout and the content on the various pages separate from one another, thereby making it easy to modify the entire site's content. The remainder of this article illustrates how to implement this example site layout using MasterPages in ASP.NET 2.0.

**Creating a MasterPage in Visual Studio 2005**  
When creating a new Web site in Visual Studio 2005, one of the first things you'll likely want to do is create a MasterPage. To create a new MasterPage, you simple add a new item to the Project, which will bring up the Add New Item dialog box (shown below). From this dialog box simply choose to add an item of type MasterPage. As you can see in the screenshot below, MasterPages have the default extension of .master.



Note that when adding a MasterPage you are asked to select the language for the MasterPage (Visual Basic, C#, or J#). As you might have guessed, MasterPages can support server-side code. You can add code in the Page\_Load event handler, for example. This is useful if you want the MasterPage to have some content that should not be editable on a page-by-page basis, but is dynamic nevertheless. For example, you might want to display the name of the currently logged in user in the upper-right hand corner; or you might want to display the number of currently online users in the footer.

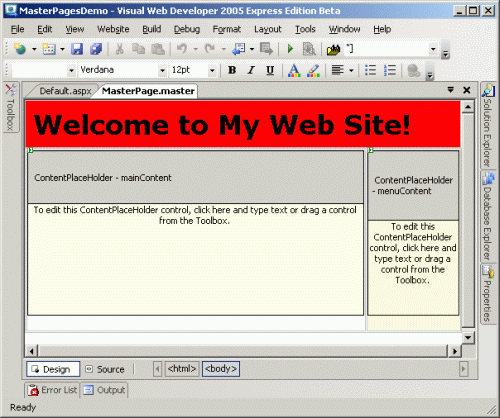
After creating a new MasterPage, you can add content to the MasterPage through the Designer (shown in the screenshot below) or declaratively. As the screenshot below shows, a new MasterPage contains nothing but a single content placeholder. Recall that the content placeholder is the portion of the MasterPage that will be supplied on a page-by-page basis.



A MasterPage can have multiple content placeholders. To add a new content placeholder in the Designer, simply drag and drop the ContentPlaceholder control from the Toolbox. You can also add a ContentPlaceholder declaratively in the HTML view. ContentPlaceholders, like any other Web control, have a declarative syntax, as shown below:

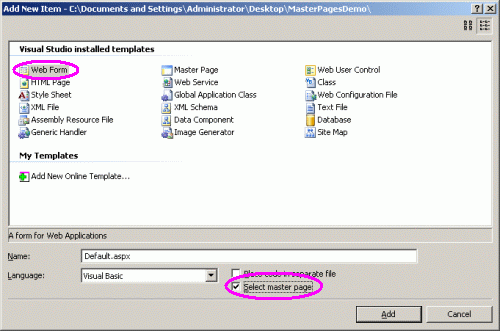
|  |
| --- |
| <asp:contentplaceholder id="mainContent" runat="server">  </asp:contentplaceholder> |

In order to create a design similar to the one we examined earlier, I crafted the MasterPage to have two content placeholders: one for the main content on the left, and one for the menu on the right. The masthead was added statically to the MasterPage. When all was said an done, I ended up with the MasterPage shown in the screenshot below:

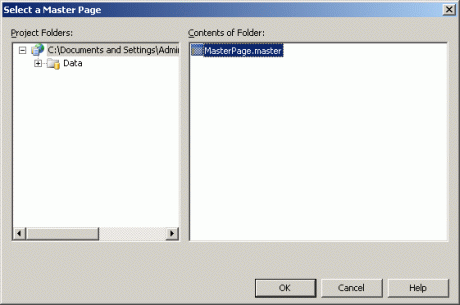


As you can see, the masthead is static and reads "Welcome to My Web Site!" Following that are two content placeholders: mainContent, the content placeholder on the left, and menuContent, the content placeholder on the right. At this point we have completed our MasterPage. As we'll see next, when creating new ASP.NET pages we can specify that the ASP.NET page should utilize a particular MasterPage. From there, we can modify the particular editable content placeholders in the page.

**Creating an ASP.NET Page Based on a MasterPage**  
Once a MasterPage has been created, when creating new ASP.NET pages you'll want to specify that these pages should inherit the layout of the MasterPage. When adding a new Web page to the Project, you can indicate that it utilize a particular MasterPage, as shown in the screenshot below:



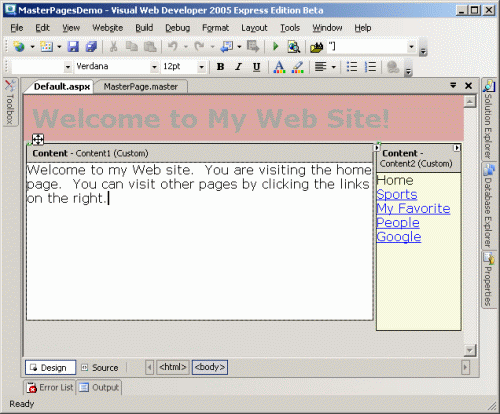
If you check the "Select master page" checkbox you'll be taken to a dialog box that prompts you to select the MasterPage that the ASP.NET page should use as a base for its layout. The screenshot below shows this dialog box. Since we have only created a single MasterPage - MasterPage.master - there is only one MasterPage available in the list. However, a site can have any number of MasterPages, as you might want different layouts for different portions of your site.



Once you have selected a MasterPage, the new ASP.NET page is created. The content of the ASP.NET page is very minimalistic - rather than having the boilerplate HTML markup that a new ASP.NET page without a MasterPage has, our new ASP.NET page simply contains a reference to the MasterPage selected:

|  |
| --- |
| <%@ Page Language="VB" MasterPageFile="~/MasterPage.master" Title="Untitled Page" %> |

That's all there is in the entire ASP.NET Web page! If you go to the Design view, though, you'll see the entire MasterPage layout. From the Design view you can add controls and markup to the content placeholders, but the markup outside of the content placeholders is dimmed and is not editable through the page. The screenshot below shows the ASP.NET page after some content has been entered into the content placeholders.



After you have added content to the ASP.NET page, when switching back to the HTML view you'll see that new Content controls have been added, containing the controls and markup you specified through the Designer. After typing in some content into both of the content placeholders, the ASP.NET page's declarative syntax looked like:

|  |
| --- |
| <%@ Page Language="VB" MasterPageFile="~/MasterPage.master" Title="Untitled Page" %>  <asp:Content ID="Content1" ContentPlaceHolderID="mainContent" Runat="server">Welcome  to my Web site.  You are visiting the home page.  You can visit other  pages by clicking the links on the right.</asp:Content>    <asp:Content ID="Content2" ContentPlaceHolderID="menuContent" Runat="server">Home<br />  <a href="Sports.aspx">Sports</a><br />  <a href="FavFolks.aspx">My Favorite People</a><br />  <a href="http://www.google.com/">Google</a></asp:Content> |

Notice how using MasterPages separates the layout and the content of the ASP.NET page. The HTML that lays the page out is separated into the MasterPage; the ASP.NET page itself contains only the page-specific content. What this separation provides is the ability to alter the entire site's layout and style by just editing the MasterPage. That is, if you want to alter the text in, say, the masthead, you just need to edit the MasterPage.master file. After making this change, any user that visits an ASP.NET page that utilizes this MasterPage will see the new text.

**Conclusion**  
In this article we took a sneak peak at MasterPages in ASP.NET 2.0. MasterPages are not a new concept, as there has been MasterPage support in ASP.NET 1.x for some time now. What is exciting about MasterPages in ASP.NET 2.0, though, is that Visual Studio 2005 introduces rich, design-time support for MasterPages, something that was sorely missing in Visual Studio .NET. With MasterPages you can achieve true separation between the markup to lay out your pages and the actual content-specific markup. This makes it easy to apply a consistent look and feel to all pages in a site and makes it a breeze to update the site-wide style.