

# Intermediate Web Development with ASP.NET

Instructor: Frank Stepanski

## Overview

From this class, you will learn more in depth features of ASP.NET and C# to create interactive, customizable, and dynamic websites.

## Audience for this Class

I will assume you have either taken my [Beginning Web Development with ASP.NET](#) class or have an understanding of the basics of ASP.NET and C# in using Visual Web Developer Express or Visual Studio as well as experience in basic web design (HTML and CSS).

I will be using [Visual Web Developer Express 2008](#) for this class but you can use earlier edition(s) or the commercial VS product too.

## Master Pages

Almost all professionally done websites have a consistent look and feel across their pages. The common look and feel usually include both the layout of the page --- where various user interface elements appear and how they are oriented with respect to one another and the fonts, colors and graphics used within the page.

For example, if you visit my favorite web site [ESPN.com](#), you'll find that regardless of where you go on the site, at the top of the page you see the logo, a search box, and a menu listing the sports covered by ESPN as well as similar colors.

When you are building web pages for a site, it is important to not add the common site-wide features manually to each and every page. That is, the web designers at ESPN.com would be inconsistent if they had to manually add the search box, logo and common links to each and every page. Not only would this add significantly to the time required to build a page, but it would make updating the websites layout a nightmare.

The approach used by professional web designers/developers is to create some sort of website template that has the HTML content that should be present on each and every page. That is accomplished in ASP.NET through the use of a **master page**.

A master page is a special ASP.NET page that, like regular ASP.NET pages, contains both a code portion and an HTML portion; with the HTML portion consisting of both static HTML content and Web controls. This master page contains the common layout content. Then, when you're creating a new ASP.NET page, this new page can be assigned to the master page, thereby inheriting its look and feel. With the master page model, updating the single master page automatically updates the common content in all of the pages that inherit that master page. With master pages, creating a unified site layout that can be easily updated is a snap!

With ASP.NET 3.5 (or older 2.0) and Visual Web Express, a page developer provides a site wide template by creating a master page. Somewhat similar to template files in [Microsoft Expression Web](#) and/or [Adobe Dreamweaver](#), a master page consists of two pieces: content that appears on each and every page that inherits the master page and regions that can be customized by pages using the master page.

The master page model in ASP.NET outshines simple HTML-based templates because master pages, like any ASP.NET page, can contain not only HTML markup, but web controls and server-side source code (C#) as well.

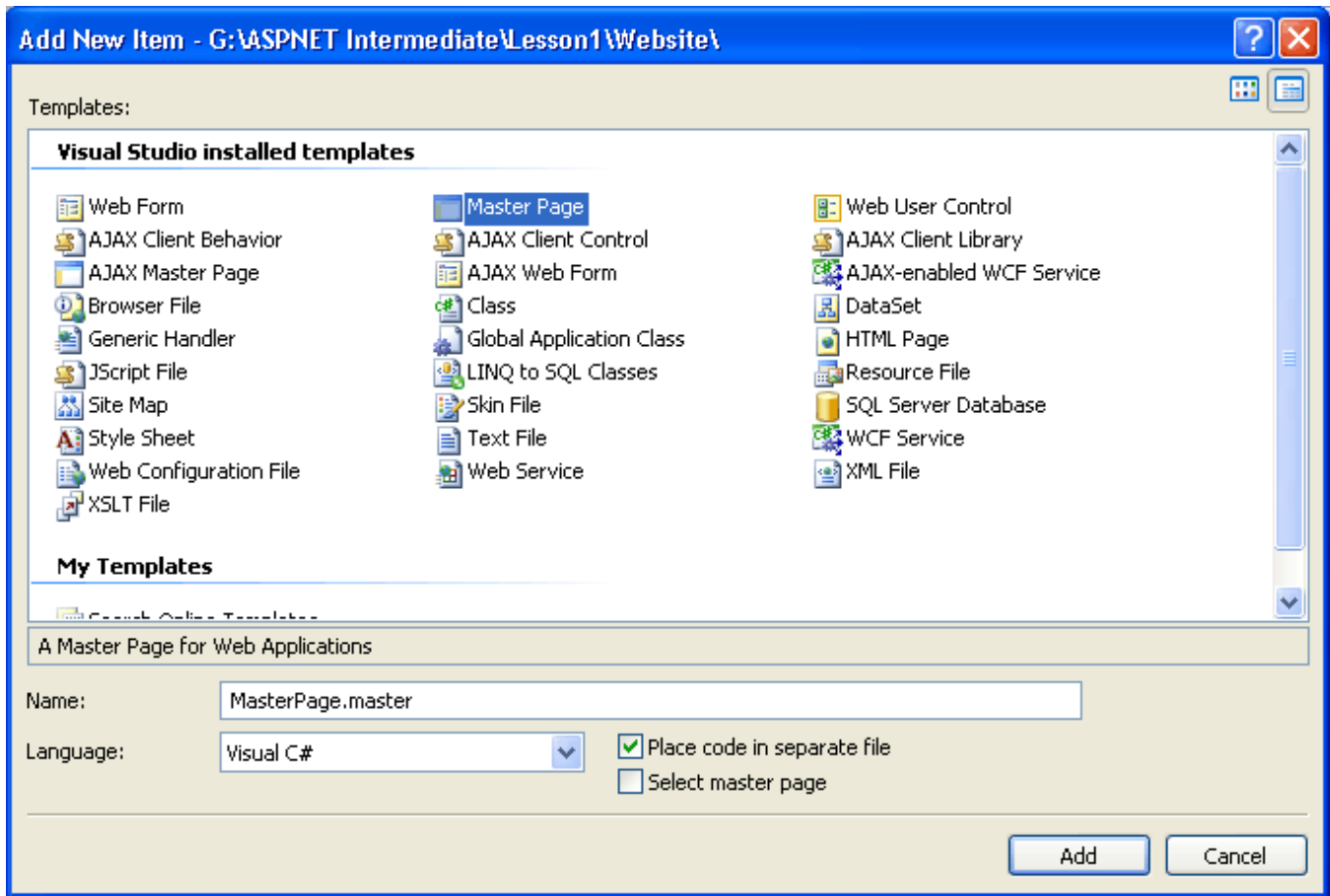
Master pages have a different file extension than a normal ASP.NET page (.master instead of .aspx), and they can't be viewed directly by a browser. Instead, master pages must be used by other pages, which are known as content pages. Essentially, the master page defines the page structure and the common ingredients. The content pages adopt this structure and just fill it with the appropriate content.

## **A Simple Master Page and Content Page**

To see how this works, let's create a very simple example.

To create a master page:

Select Website > Add New Item from the menu and Select Master Page, give it a file name, and click OK. (Figure 1)



**Figure 1 – Adding a Master Page**

When you create a new master page in Visual Web Developer, you start with a blank page that includes a single [ContentPlaceHolder control](#) (Figure 2).

```

13 | <form id="form1" runat="server">
14 | <div>
15 |     <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
16 |
17 |     </asp:ContentPlaceHolder>
18 | </div>
19 | </form>

```

**Figure 2 – Source view of ContentPlaceHolder control in Master Page**

The ContentPlaceHolder is the portion of the master page that has content that can change. Or, to look at it another way, everything else in a master page is unchangeable.

If you add a header, that header appears in every content page. If you want to give the content page the opportunity to supply content in a specific section of the page, you need to add a ContentPlaceHolder (s).

When you first create a master page, you'll start with nothing more than a single ContentPlaceHolder. To make this example a little more practical, try adding a header before the ContentPlaceHolder and a footer after it (and maybe a little styling too), as shown in Figure 3, 3a and 3b.

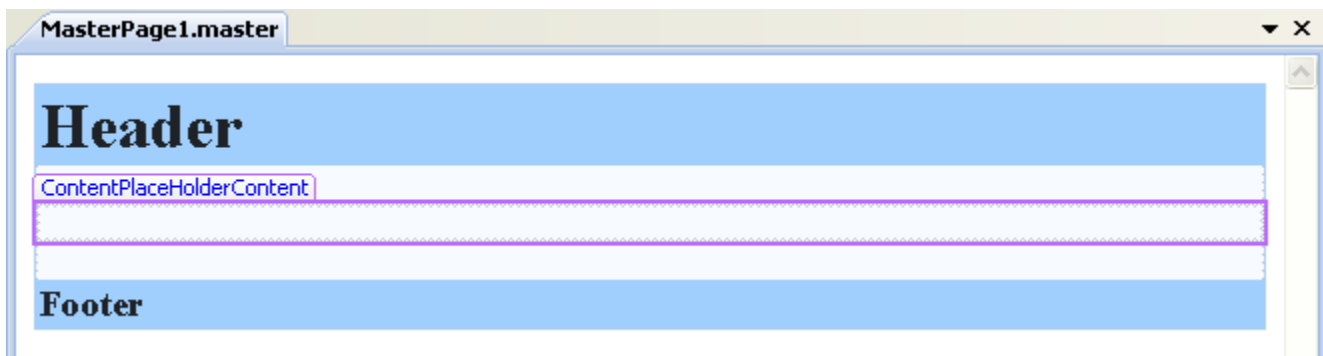
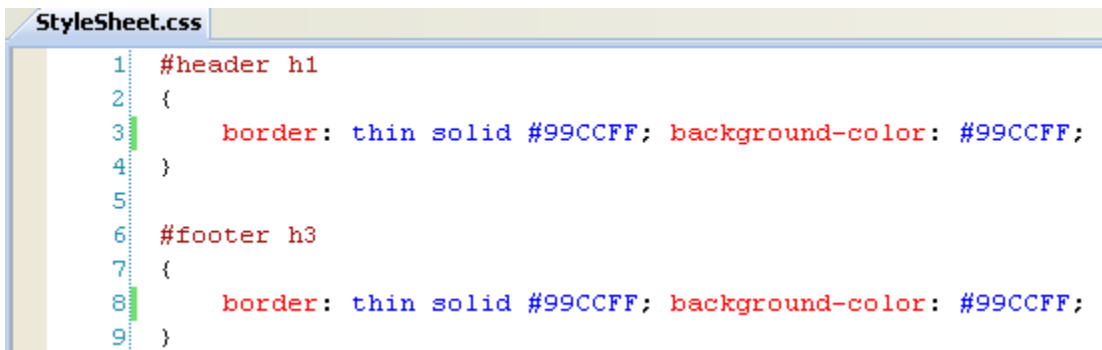


Figure 3 – Adding a very simple header and footer to the Master Pager

```
12 | <form id="form1" runat="server">
13 | <div>
14 |
15 | <div id = "header"><h1>Header</h1></div>
16 |
17 | <asp:ContentPlaceHolder id="ContentPlaceHolderContent" runat="server">
18 | </asp:ContentPlaceHolder>
19 |
20 | <div id = "footer"><h3>Footer</h3></div>
21 |
22 | </div>
23 | </form>
```

**Figure 3a – Source view of Master Page**

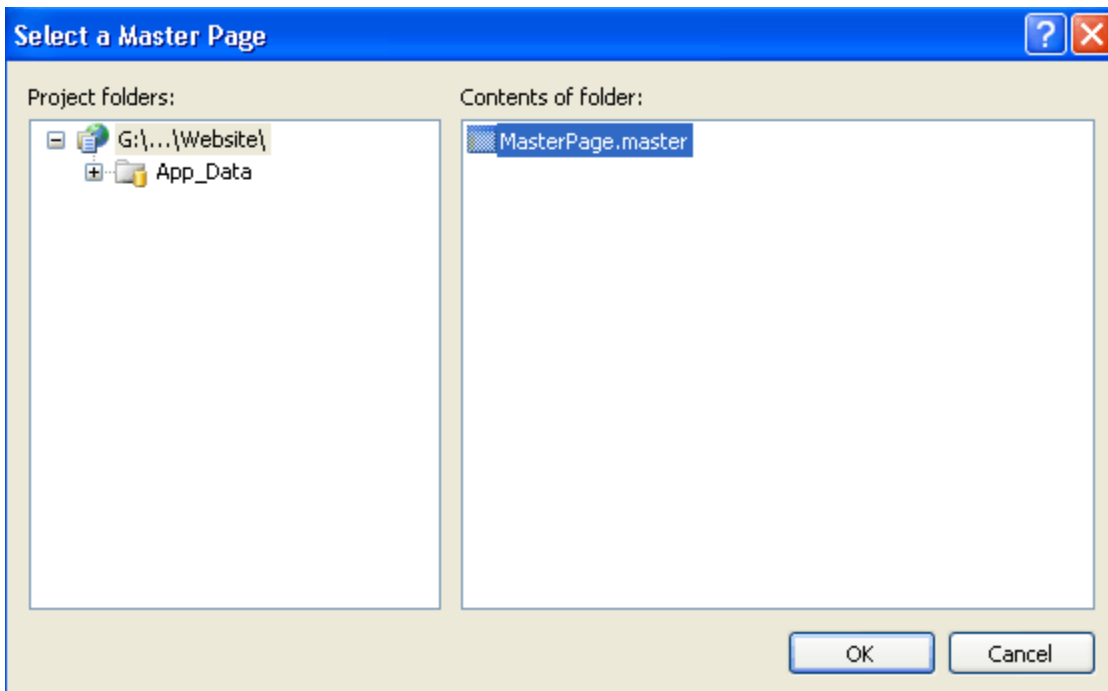


```
1 #header h1
2 {
3     border: thin solid #99CCFF; background-color: #99CCFF;
4 }
5
6 #footer h3
7 {
8     border: thin solid #99CCFF; background-color: #99CCFF;
9 }
```

**Figure 3b – Style sheet for Master Page**

Now you're ready to create a content page (.aspx) based on this master page.

Select Website > Add New Item from the menu then Select Web Form, and choose a master page (Figure 4). Click OK. When you're prompted to choose a master page, use the one you created with the header and footer.



**Figure 4 – Add a web form to use Master Page**

Now you'll see something a little more interesting. Your content page will have all the elements of the master page, but the elements will be unavailable to change, indicating you can't select or change them in any way.

The only editable area will be within the ContentPlaceHolder control.

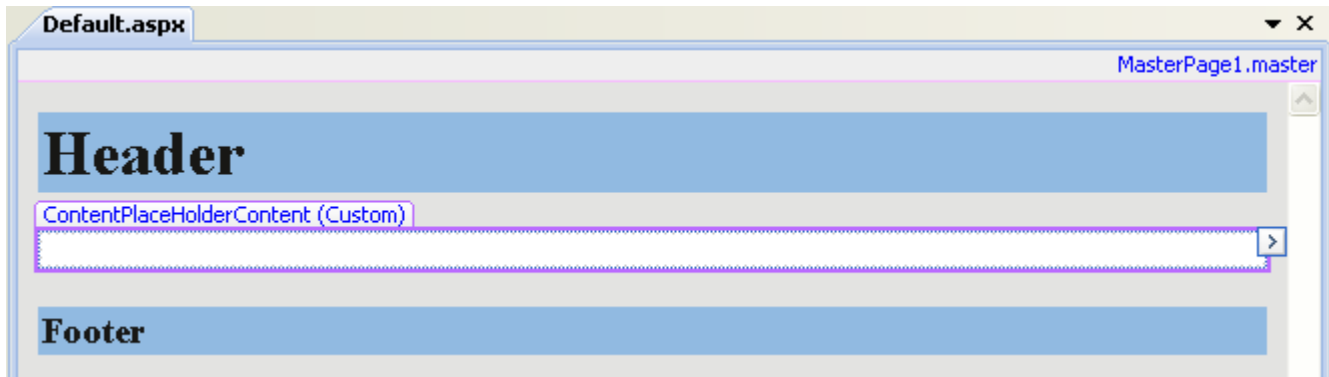


Figure 5 – Content Page using a Master Page

```
1 <%@ Page Language="C#" MasterPageFile="~/MasterPage1.master"
2 AutoEventWireup="true" CodeFile="Default.aspx.cs"
3 Inherits="_Default" Title="Untitled Page" %>
4
5 <asp:Content ID="Content1"
6 ContentPlaceHolderID="ContentPlaceHolderContent"
7 Runat="Server">
8
9 <!-- content goes here -->
10
11 </asp:Content>
```

Figure 5a– Source view of Default.aspx (Content Page)

**My version:**

[http://fsjay7123.brinkster.net/ASPNET\\_Intermediate/Lesson1/Default.aspx](http://fsjay7123.brinkster.net/ASPNET_Intermediate/Lesson1/Default.aspx)

The real magic starts when you create multiple pages (.aspx) that use the same master page. Now, each page will have the same header and footer, creating a seamless look across your entire website.

**Note:** Don't worry I will be using a more "real" web design site example as we continue to work with master pages in the upcoming lessons. 😊

## How Master Pages and Content Pages Are Connected

Now that you've seen a master page example, it's worth taking a look behind the scenes to see how you implement the master page.

When you create a master page, you're building something that looks much like an ordinary ASP.NET web form. The key difference is that although web forms start with the Page directive, a master page starts with a Master directive that specifies the same information.

Here's the Master directive for the simple master page shown in the previous example:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage1.master.cs" Inherits="MasterPage" %>
```

When you create a content page, ASP.NET links your page to the master page by adding an attribute to the Page directive. This attribute, named MasterPageFile, indicates the associated master page.

Here's what it looks like:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage1.master" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" Title="Untitled Page" %>
```

**Note:** By default, the MasterPageFile attribute begins with the path `~/` to specify the root website folder. This will work fine locally, but when you upload it to your web host (i.e. Brinkster) you probably will have to take the `~/` off and it should work fine. My weekly examples already have them taken off.

The Page directive has another new attribute – Title. That’s because the master page, as the outermost shell of the page, always defines the `<head>` section of the page, which includes its title.

Remember, your content page can’t modify anything that’s in the master page. However, this is an obvious shortcoming with the title information, so to circumvent it, ASP.NET adds the *Title* attribute, which you can set to override the title specified in the master page with something more appropriate. This system works as long as the master page has the `runat="server"` attribute in the `<head>` tag, which is the default.

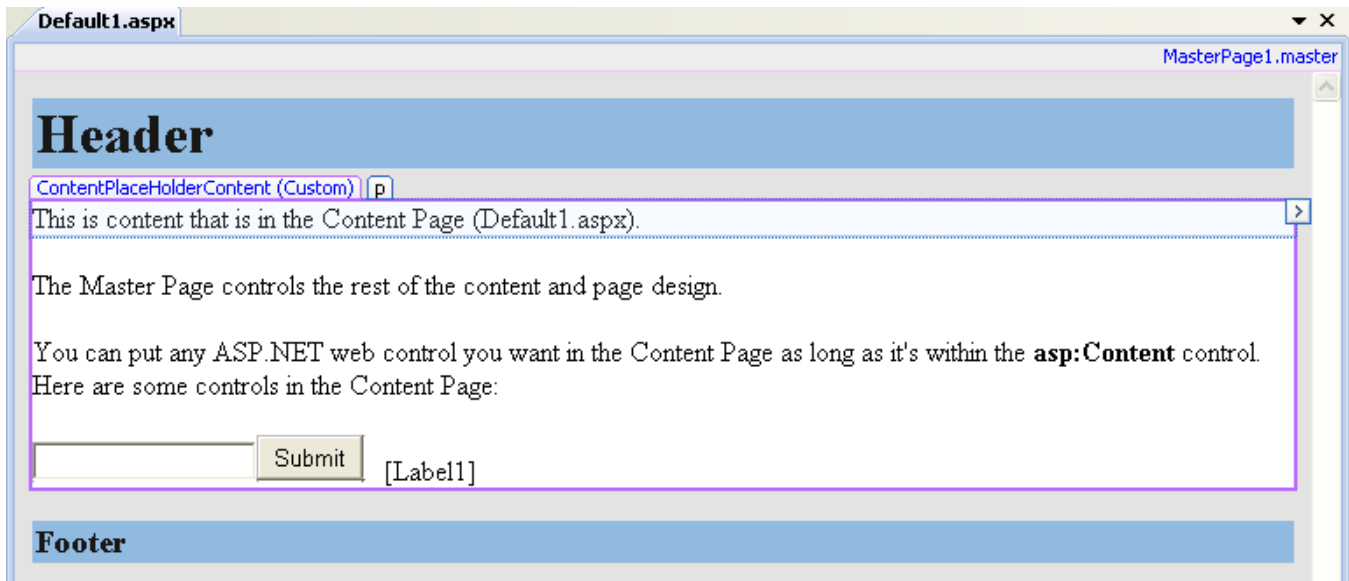
**Note:** I’ll cover how to change the title on each of your content pages in the next lesson.

The rest of the content page looks a little different from an ordinary web form. That’s because the content page can’t define anything that’s already provided in the master page, including the `<head>` section, the root `<html>` element, the `<body>` element, and so on.

In fact, the content page can do only one thing – it can supply a Content tag that corresponds to the ContentPlaceHolder in the master page. This is where you insert the content for this page. As a result, your content pages are a little bit simpler than ordinary web pages.

For ASP.NET to process this page successfully, the **ContentPlaceHolderID** attribute in the `<Content>` tag must match the **ContentPlaceHolder** specified in the master page exactly. This is how ASP.NET knows where it should insert your content in the master page template.

Here’s our content page looks now with some content in it (Figure 6):



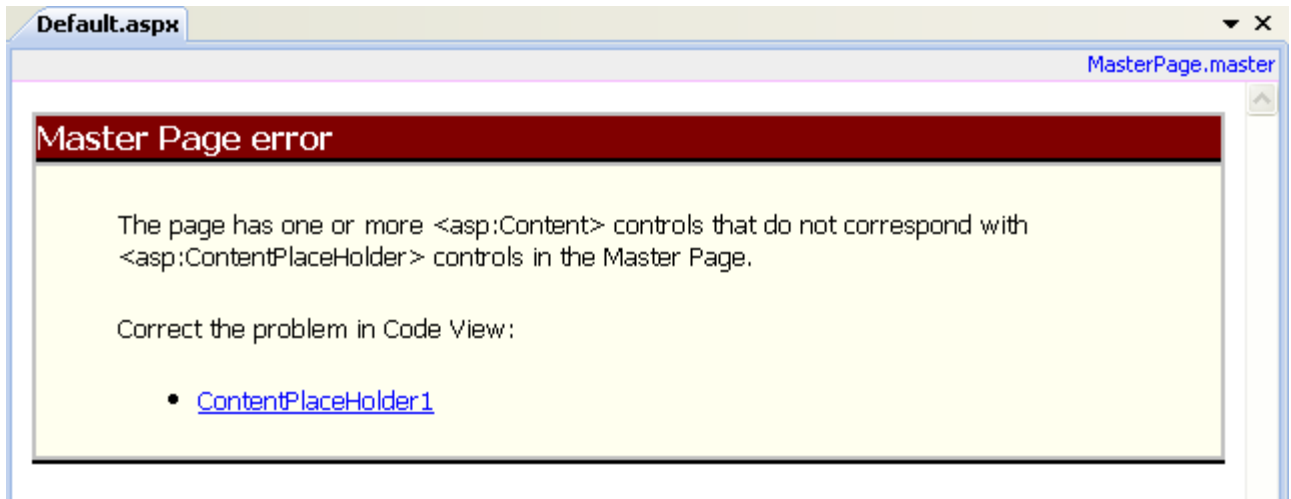
**Figure 6 – Design View of Content Page with some content**

**My version:**

[http://fsjay7123.brinkster.net/ASPNET\\_Intermediate/Lesson1/Default1.aspx](http://fsjay7123.brinkster.net/ASPNET_Intermediate/Lesson1/Default1.aspx)

You'll notice that Visual Web Developer 2008 very nicely (improvement from 2005 version) shows the developer exactly what ContentPlaceHolder control has in the content. Everything within that purple outline contains the content in that control and everything outside of that will be what the master page controls.

**Note:** If you change the names of the ContentPlaceHolder controls in your master page, make sure you match those changes in the content pages. Otherwise you'll get an error in the Design View of you content pages like this (Figure 7):

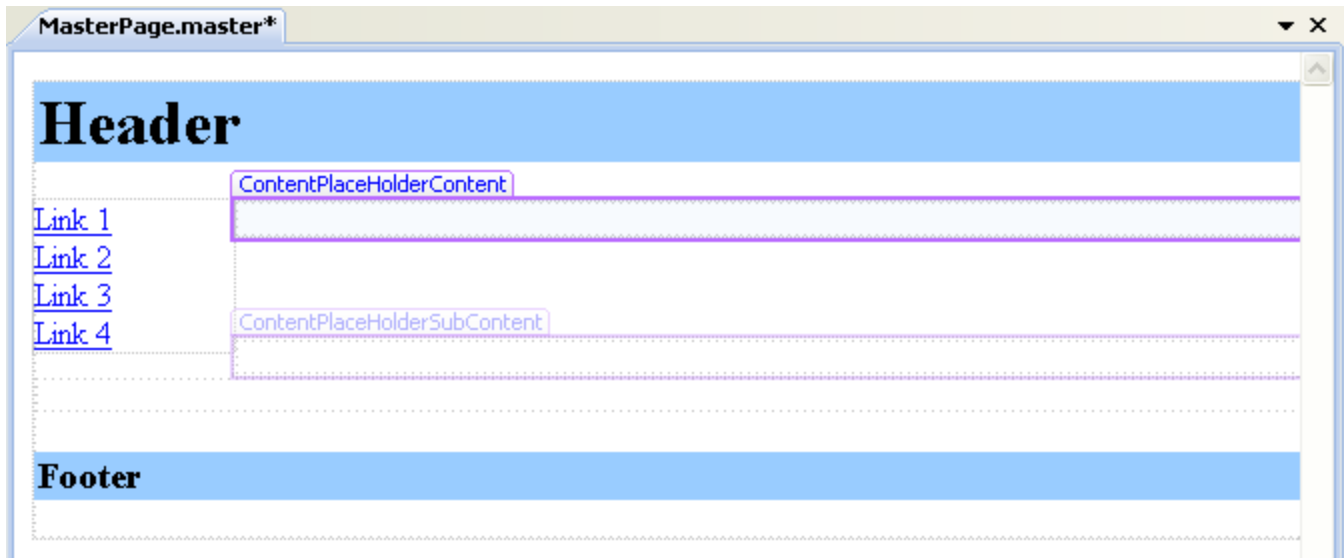


**Figure 7 – Non-matching ContentPlaceHolder control in Content Page**

## **A Master Page with Multiple Content Regions**

Master pages aren't limited to one ContentPlaceHolder. You can insert as many as you need to give the designer/developer the ability to intersperse content in various places. All you need to do is add multiple ContentPlaceHolder controls and arrange them appropriately.

**Figure 8** shows a modified master page from the previous example showing you a somewhat real design for a simple web site.



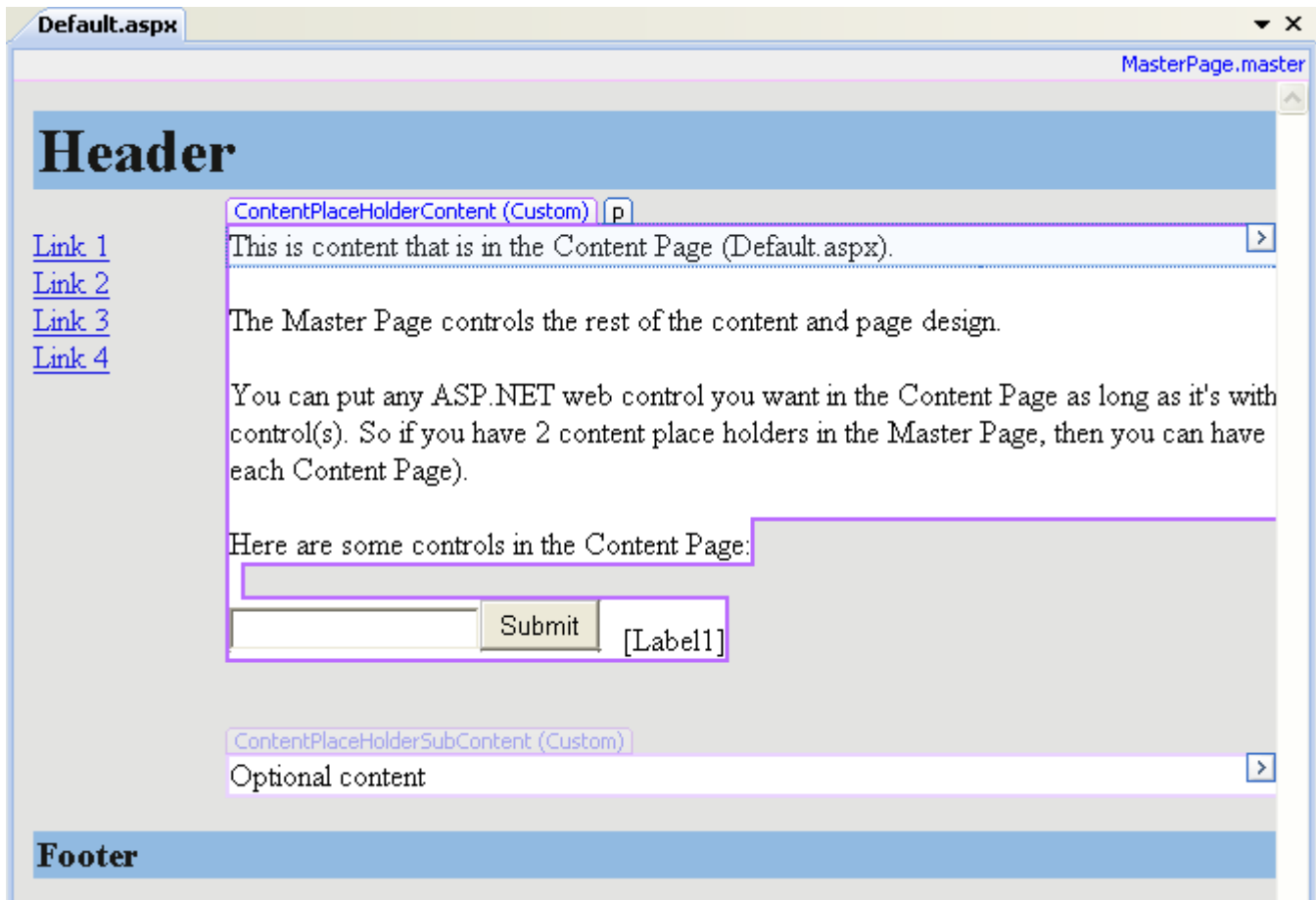
**Figure 8 – Updated Master Page to simulate a real web design**

I wanted to finish up this lesson showing you a more practical design that you can use. I changed the styles to lay out a simple left navigation and floated the main content sections to the right. Feel free to review the stylesheet.css to see how I designed it.

**Note:** One prerequisite for this class is a general understanding of web design (which includes using basic style sheets). Feel free to ask me CSS question(s), but this class is covering web development not web design. I will leave the web design to the designers, which I am not. 😊

When you look at this new master page, you can see I included a second ContentPlaceholder for additional content. You can have multiple place holders for content that your content pages can use, but don't think that every page must use them.

Now here is how a new content page will look using this new master page (Figure 9):



**Figure 9 – Content Page using updated Master Page**

**My version:**

[http://fsjay7123.brinkster.net/ASPNET\\_Intermediate/Lesson1/Default2.aspx](http://fsjay7123.brinkster.net/ASPNET_Intermediate/Lesson1/Default2.aspx)

**Note:** You may have noticed that there is no ContentPlaceHolder control in the toolbox to physically place on your pages. The reason is because every content control (*asp: content*) declaration is automatically created when you specify what master page you will use (Figure 4) and it is only applicable if you are using a master page as opposed to the controls on the toolbox which can be used for any .aspx page.

So if you go back to your master page and add a new ContentPlaceHolder, you can copy and paste the declaration and change the name it references in the master page.

## Brinkster Web Host Support

<http://www.brinkster.com/hosting/FreeDeveloper.aspx>

The free web hosting developer package that is offered by Brinkster does not support the latest Framework of ASP.NET (version 3.5). So you will need to make some slight changes to all the .cs pages when you upload them to Brinkster.

I've already made these changes in the examples I've included in the weekly .zip files, but here is what the changes look like (Figure 10):

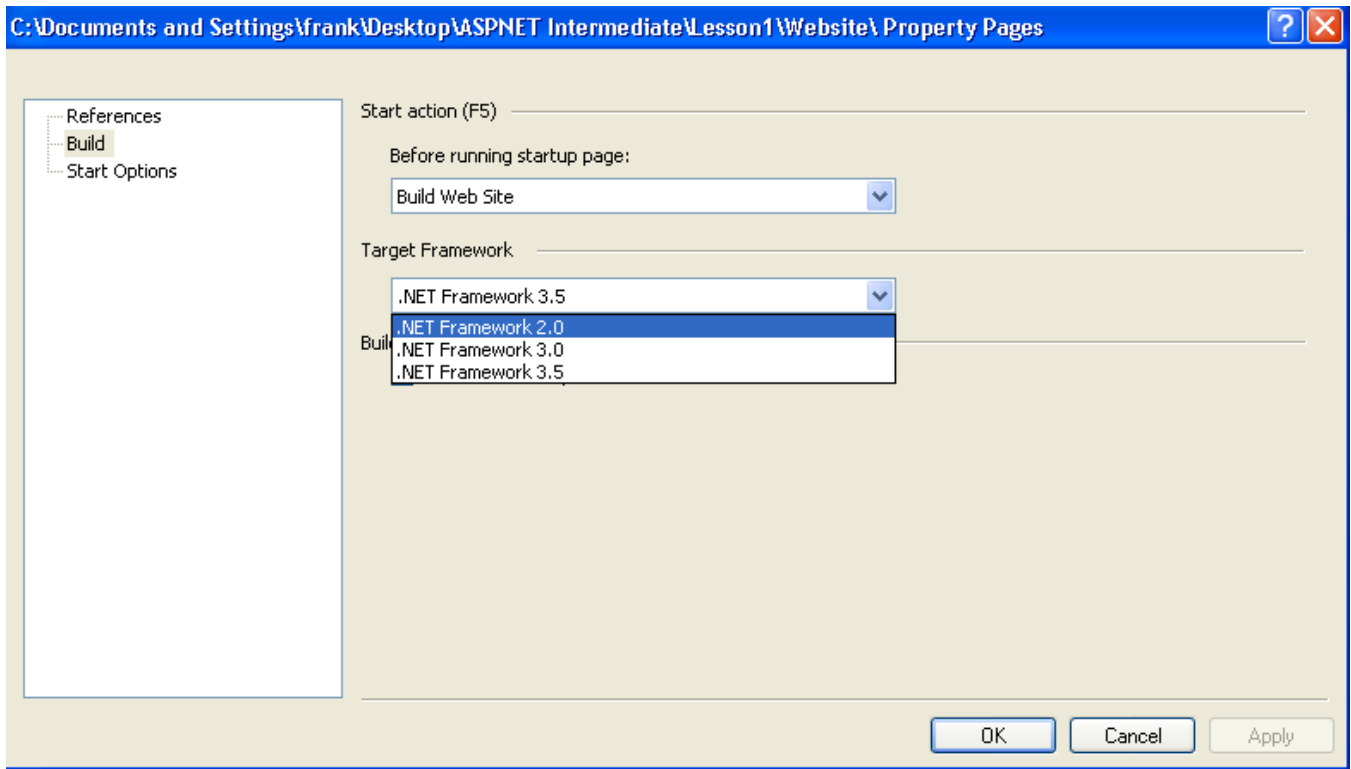
```
1 using System;
2 using System.Collections;
3 using System.Configuration;
4 using System.Data;
5 using System.Web;
6 using System.Web.Security;
7 using System.Web.UI;
8 using System.Web.UI.HtmlControls;
9 using System.Web.UI.WebControls;
10 using System.Web.UI.WebControls.WebParts;
```

**Figure 10 – Changes to the libraries that are supported by Brinkster's free account.**

**Note:** Now if you have your own web host that fully supports ASP.NET 3.5 or if you don't need to upload your assignments then you can disregard.

Any further questions on this please let me know. 😊

You can also change the .NET Framework that Visual Web Developer builds and will be distributed by right-clicking on your website and selecting the Property Pages of the site (Figure 11):



**Figure 11 – Property Page for website to change the .NET Framework of site**

## Assignment for Lesson 1

1. Setup your web host if you do not already have one.
2. Review the lesson on the basics of master pages and create a simple master page (.master) and content page (.aspx) of your own.
3. Post on the class discussion board that your Lesson 1 is ready with a direct link to your page, such as:

<http://fsjay7123.brinkster.net/frank-lesson1.aspx>

I will respond with a review on the class discussion board.

Copyright 2009 © Frank Stepanski

Used with Permission :: LVS Online Classes / LVS Associates

Lessons, files and content of these classes cannot be reproduced and/or published without the express written consent of the author.

Use of this site implies agreement with the [Terms of Use](#)