SOFTWARE WORKSHOP

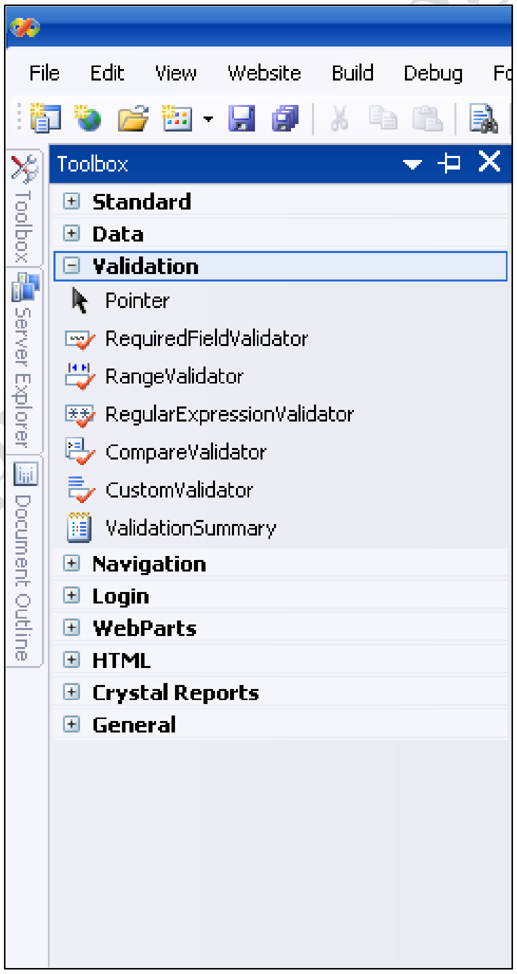
- Lab Exercise
- Study Material

Validation Controls

Validations are used on the fields of a form in ASP.NET webpage. They are used to make restrictions when entering the fields .

For ex- We want that first name is required field while signing up then it would be valid only when it contains some value.

If we perform all such checks it becomes a tedious task. So ASP.NET has in-built functionality for this called Validation controls.



You can see these under 'Validations' headings in Toolbox

The six types of validations: RequiredFieldValidator, RangeValidator, RegularExpressionValidator, CompareValidator, CustomValidator, and ValidationSummary.

<u>To understand the concept perform following steps :</u>

1. **Open a new ASP.NET website.**

2. **On Default.aspx add one textbox and one button.**

3. **Drag "RequiredFieldValidator" from Validations headings in Toolbox and drop it to your page.**

4. **Right Click on "RequiredFieldValidator" and go on its properties.**

5. **See the 'ControlToValidate' property – In this you will be selecting the control for which you want validation. Select your textbox here.**

6. **Now your text box has become a Required Field, thus user have to fill it.**

7. **Now run your project and see the output.**

<u>Common validation control properties</u>(Applicable to all) VIMP

Property	**Description**
ControlToValidate	The name of the control to be validated.
Error Message	The message that's displayed in the validation control when the validation fails.
Text	The message that will appear beside the field when validated is the text.

This was about RequiredFieldValidator lets see other validation controls.

Range Validator

Additional properties of a range Validator

Property	Description
Maximum Value	The maximum value that the control can contain.
Minimum Value	The minimum value that the control can contain.
Type	The data type to use for range checking (String, Integer, Double, Date, or Currency).

Regular Expression Validator

The regular expression validator is used to validate an asp.net control to an expression the user specifies. The most popular example is validating the email format of the field. The important properties are below.

Ex- If email-id is to be given in one box, then its format must match it.

Property	Description
Validation Expression	This is the place where you choose the standard expression you wish to use in your validation

Compare Validator

The main two essential roles for the compare validator are to compare the entry of 2 fields in a web form and to check data type of the data entered in the field. The important properties are below.

This is mainly used in Password : / Confirm Password thing.

Property	Description
ControlToCompare	You can choose the ID of the control you want to compare with the ID of the control you set in the ControlToValidate property.

CustomValidator

The custom validator is used to create your validation function and set to the custom validator. This will go somewhat complex. So, we will talk about it in lab discussion.

ValidationSummary

The validation summary displays the list of all the messages that you set in each validator when the validation fires. The most important properties are the following.

Property	Description
Displaymode	The layout the messages that will display on is the Displaymode.
ShowMessageBox	If set to true, a popup alert window will open displaying all the validation messages of the control.
ShowSummary	If set to true, the validation messages will display on the page.
Text	The message that will appear beside the field when validated is the text.

Task I (30 min.)

Create asp.net website illustrating usage of all the validation controls.

WORKING WITH MASTER PAGES

One attribute of a well-designed website is a consistent site-wide page layout. Sites have same headers, footers in all their pages.

Another attribute of a well-designed site is the ease with which the site's appearance can be changed. Applying changes to the entire site should be a fast and simple process that does not require modifying the thousands of web pages that make up the site.

Creating a site-wide page template in ASP.NET is possible through the use of *master pages*. In a nutshell, a master page is a special type of ASP.NET page that defines the markup that is common among all *content pages* as well as regions that are customizable on a content page-by-content page basis. (A content page is an ASP.NET page that is bound to the master page.) Whenever a master page's layout or formatting is changed, all of its content pages' output is likewise immediately updated, which makes applying site-wide appearance changes as easy as updating and deploying a single file (namely, the master page).

Over the course of this tutorial series we:

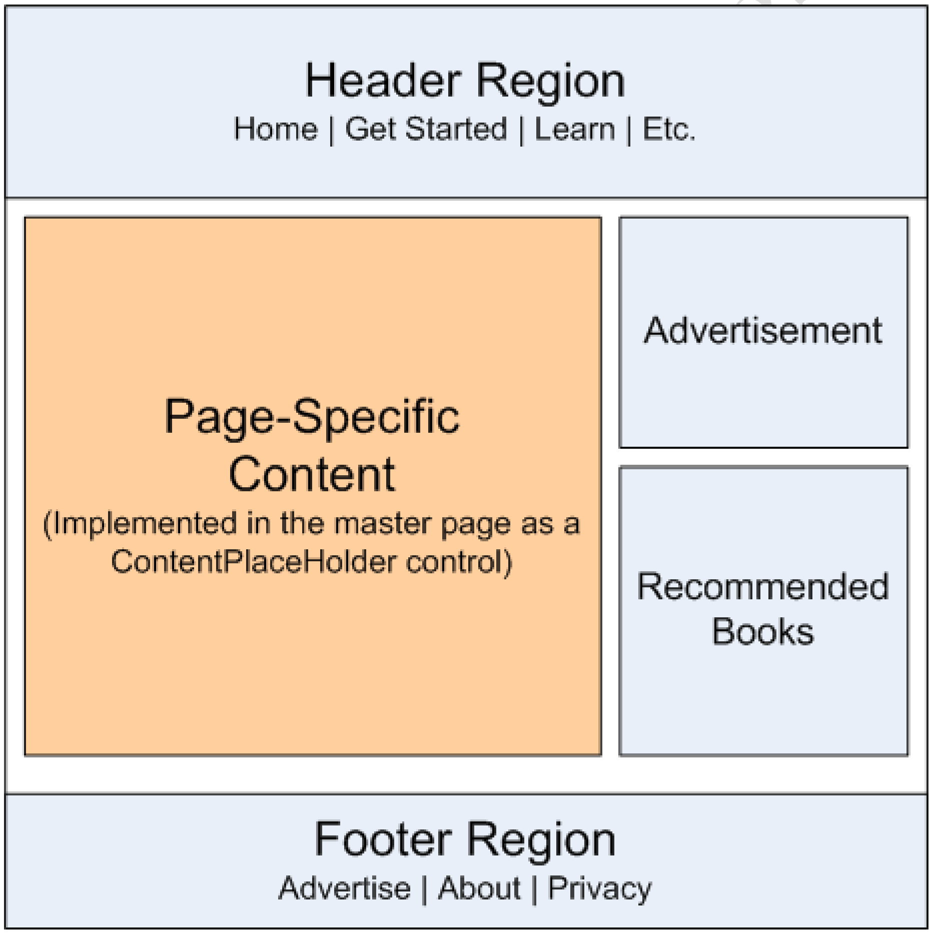
- Examine creating master pages and their associated content pages,
- Discuss a variety of tips, tricks, and traps,
- Identify common master page pitfalls and explore workarounds,
- See how to access the master page from a content page and vice-a-versa,
- Learn how to specify a content page's master page at runtime, and
- Other advanced master page topics.

Understanding How Master Pages Work

Building a website with a consistent site-wide page layout requires that each web page emit common formatting markup in addition to its custom content.

ASP.NET version 2.0 and Visual Studio 2005 are equipped with *master pages*. A master page is a special type of ASP.NET page that defines both the site-wide markup and the *regions* where associated *content pages* define their custom markup. As we will see in Step 1, these regions are defined by ContentPlaceHolder controls. The ContentPlaceHolder control simply denotes a position in the master page's control hierarchy where custom content can be injected by a content page.

Figure shows what the master page look like. Note that the master page defines the common site-wide layout - the markup at the top, bottom, and right of every page - as well as a ContentPlaceHolder in the middle-left, where the unique content for each individual web page is located.


Header Region
Home | Get Started | Learn | Etc.
Page-Specific Content
(Implemented in the master page as a ContentPlaceHolder control)
Advertisement
Recommended Books
Footer Region
Advertise | About | Privacy

Once a master page has been defined it can be bound to new ASP.NET pages through the tick of a checkbox. These ASP.NET pages - called content pages - include a Content control for each of the master page's ContentPlaceHolder controls. When the content page is visited through a browser the ASP.NET engine creates the master page's control hierarchy and injects the content page's control hierarchy into the appropriate places. This combined control hierarchy is rendered and the resulting HTML is returned to the end user's browser. Consequently, the content page emits both the common markup defined in its master page outside of the ContentPlaceHolder controls and the page-specific markup defined within its own Content controls. Figure below illustrates this concept.

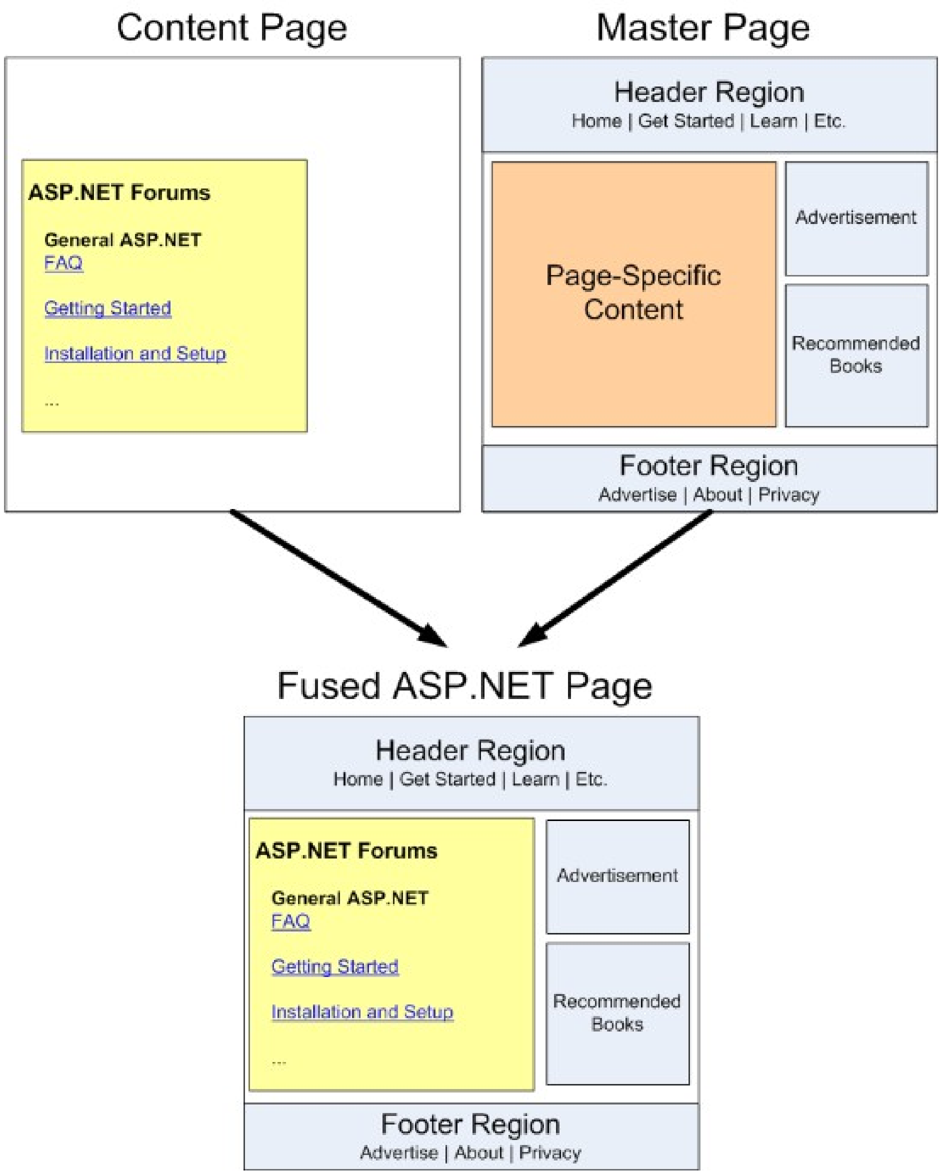
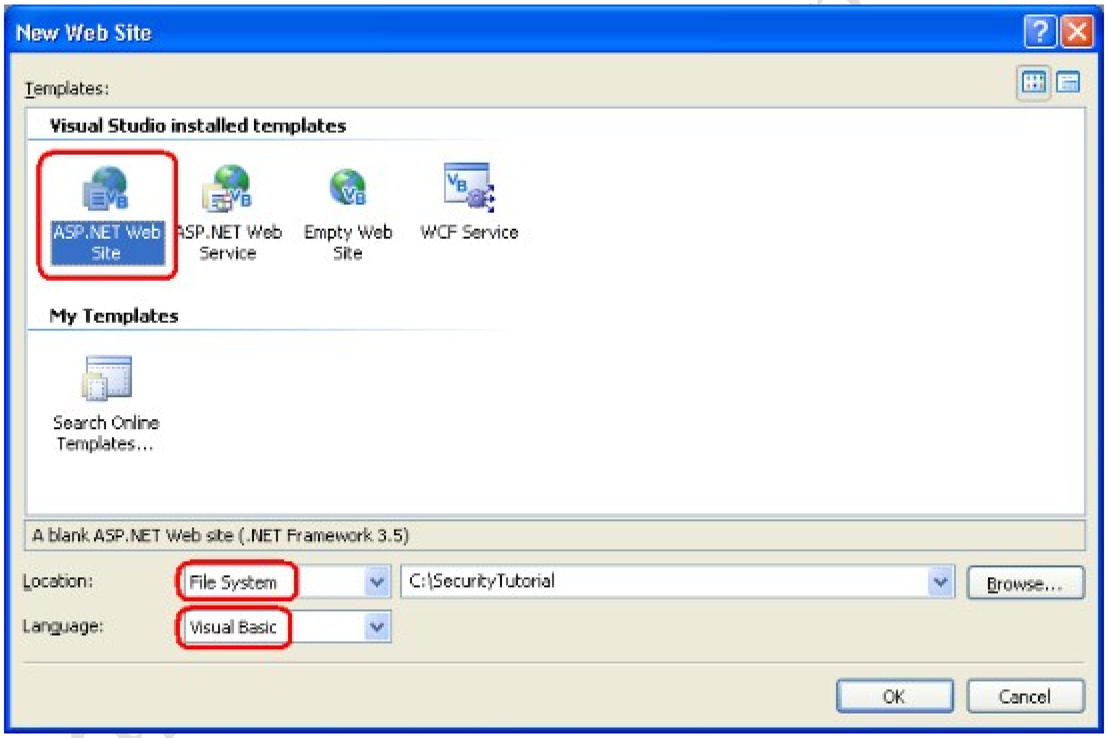


Figure : The Requested Page's Markup is Fused into the Master Page

Now that we have discussed how master pages work, let's take a look at creating a master page and associated content pages using Visual Studio 2005.

Step 1: Creating a Master Page

Before we can explore creating and using master and content pages, we first need an ASP.NET website. Start by creating a new file system-based ASP.NET website. To accomplish this, launch Visual Studio 2005 and then go to the File menu and choose New Web Site, displaying the New Web Site dialog box (see Figure below). Choose the ASP.NET Web Site template, set the Location drop-down list to File System, choose a folder to place the web site, and set the language to Visual Basic. This will create a new web site with a Default.aspx ASP.NET page, an App_Data folder, and a Web.config file.



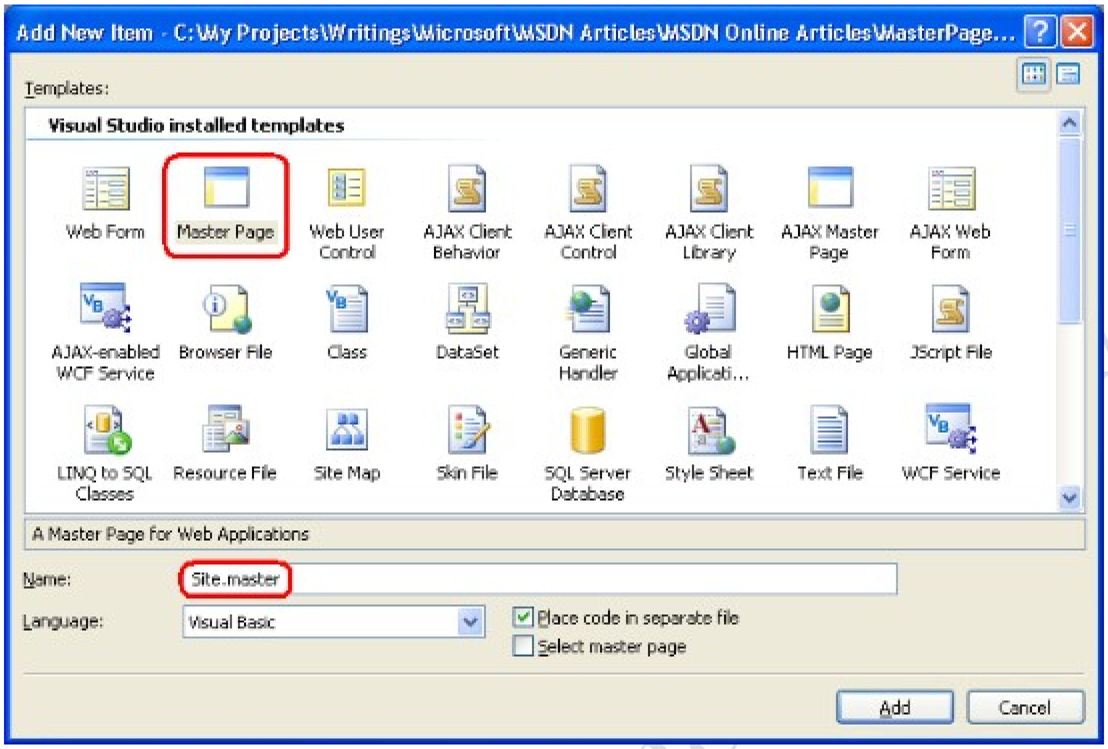
A blank ASP.NET Web site (.NET Framework 3.5)

Location: File System C:\SecurityTutorial

Language: Visual Basic

Figure : Create a New File System-Based Web Site

Next, add a master page to the site in the root directory by right-clicking on the Project name, choosing Add New Item, and selecting the Master Page template. Note that master pages end with the extension .master. Name this new master page Site.master and click Add.



A Master Page for Web Applications

Name: Site.master

Language: Visual Basic

☑ Place code in separate file

☐ Select master page

Figure : Add a Master Page Named Site.master to the Website

Adding a new master page file through Visual Studio 2005creates a master page with the following declarative markup:

<%@ 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>
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>

The first line in the declarative markup is the @Master directive. The @Master directive is similar to the @Page directive that appears in ASP.NET pages. It defines the server-side language (VB) and information about the location and inheritance of the master page's code-behind class.

The DOCTYPE and the page's declarative markup appears beneath the @Master directive. The page includes static HTML along with four server-side controls:

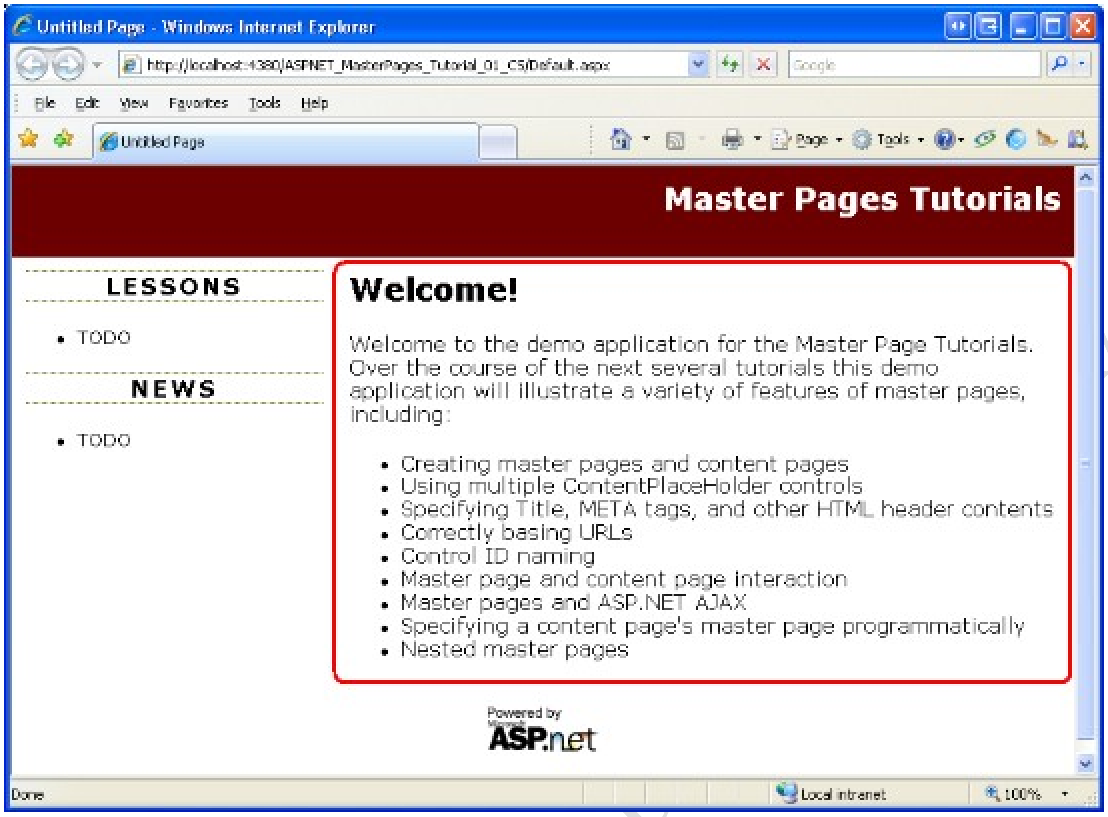
- **A Web Form (the <form runat="server">)** - because all ASP.NET pages typically have a Web Form - and because the master page may include Web controls that must appear within a Web Form - be sure to add the Web Form to your master page (rather than adding a Web Form to each content page).
- **A ContentPlaceHolder control named ContentPlaceHolder1** - this ContentPlaceHolder control appears within the Web Form and serves as the region for the content page's user interface.
- **A server-side <head> element** - the <head> element has the runat="server" attribute, making it accessible through server-side code. The <head> element is implemented this way so that the page's title and other <head>-related markup may be added or adjusted programmatically. For example, setting an ASP.NET page's Title property changes the <title> element rendered by the <head> server control.
- **A ContentPlaceHolder control named head** - this ContentPlaceHolder control appears within the <head> server control and can be used to declaratively add content to the <head> element.

This default master page declarative markup serves as a starting point for designing your own master pages. Feel free to edit the HTML or to add additional Web controls or ContentPlaceHolders to the master page.

Note: When designing a master page make sure that the master page contains a Web Form and that at least one ContentPlaceHolder control appears within this Web Form.

Creating a Simple Site Layout

Let's expand Site.master's default declarative markup to create a site layout where all pages share: a common header; a left column with navigation, news and other site-wide content; and a footer that displays the "Day of Dot Net" icon. Figure below shows the end result of the master page when one of its content pages is viewed through a browser. The red circled region in Figure is specific to the page being visited (Default.aspx); the other content is defined in the master page and therefore consistent across all content pages.


Untitled Page - Windows Internet Explorer
http://localhost:4380/ASPNET_MasterPages_Tutorial_01_CS/Default.aspx
File Edit View Favorites Tools Help
Untitled Page
Page Tools
Done
Local intranet
100%

Master Pages Tutorials

LESSONS

- TODO

NEWS

- TODO

Welcome!

Welcome to the demo application for the Master Page Tutorials. Over the course of the next several tutorials this demo application will illustrate a variety of features of master pages, including:

- Creating master pages and content pages
- Using multiple ContentPlaceHolder controls
- Specifying Title, META tags, and other HTML header contents
- Correctly basing URLs
- Control ID naming
- Master page and content page interaction
- Master pages and ASP.NET AJAX
- Specifying a content page's master page programmatically
- Nested master pages

Powered by ASP.net

Figure : The Master Page Defines the Markup for the Top, Left, and Bottom Portions

To achieve the site layout shown in Figure , start by updating the Site.master master page so that it contains the following declarative markup:

```
<%@ 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>
<link href="Styles.css" rel="stylesheet" type="text/css" />
</head> <body>
<form id="form1" runat="server">
<div id="topContent">
<a href="Default.aspx">Master Pages Tutorials</a>
</div>
<div id="mainContent">
<asp:ContentPlaceHolder id="MainContent" runat="server"> </asp:ContentPlaceHolder>
</div>
<div id="leftContent">
<h3>Lessons</h3>
<ul>
<li>TODO</li>
</ul>
```

```
<h3>News</h3>
<ul>
<li>TODO</li>
</ul>
</div>
<div id="footerContent">
<img src="Images/DayofDotNet.gif" alt="Powered by ASP.NET!" /> </div>    </form> </body>
</html>
```

The master page's layout is defined using a series of <div> HTML elements. The topContent <div> contains the markup that appears at the top of each page, while the mainContent, leftContent, and footerContent <div>s are used to display the page's content, the left column, and the "Day of Dot Net" icon, respectively. In addition to adding these <div> elements, I also renamed the ID property of the primary ContentPlaceHolder control from ContentPlaceHolder1 to MainContent.

The formatting and layout rules for these assorted <div> elements is spelled out in the Cascading Stylesheet (CSS) file Styles.css, which is specified via a <link> element in the master page's <head> element. These various rules define the look and feel of each <div> element noted above. For example, the topContent <div> element, which displays the "Master Pages Tutorials" text and link, has its formatting rules specified in Styles.css as follows:

```
#topContent {    text-align: right;    background-color: #600;    color: White;    font-size: x-large;
   text-decoration: none;    font-weight: bold;    padding: 10px;    height: 50px; }
```

If you are following along at your computer, you will need to copy this tutorial's accompanying code and add the Styles.css file to your project (on your desktop in folder /Lab Exercise). Similarly, you will also need to create a folder named Images and copy the "Day of Dot Net" icon from the downloaded demo website to your project.

Note: A discussion of CSS and web page formatting is beyond the scope of our lab. For more on CSS, check out the CSS Tutorials at W3Schools.com. I also encourage you to download this tutorial's accompanying code and play with the CSS settings in Styles.css to see the effects of different formatting rules.

Creating a Master Page Using an Existing Design Template

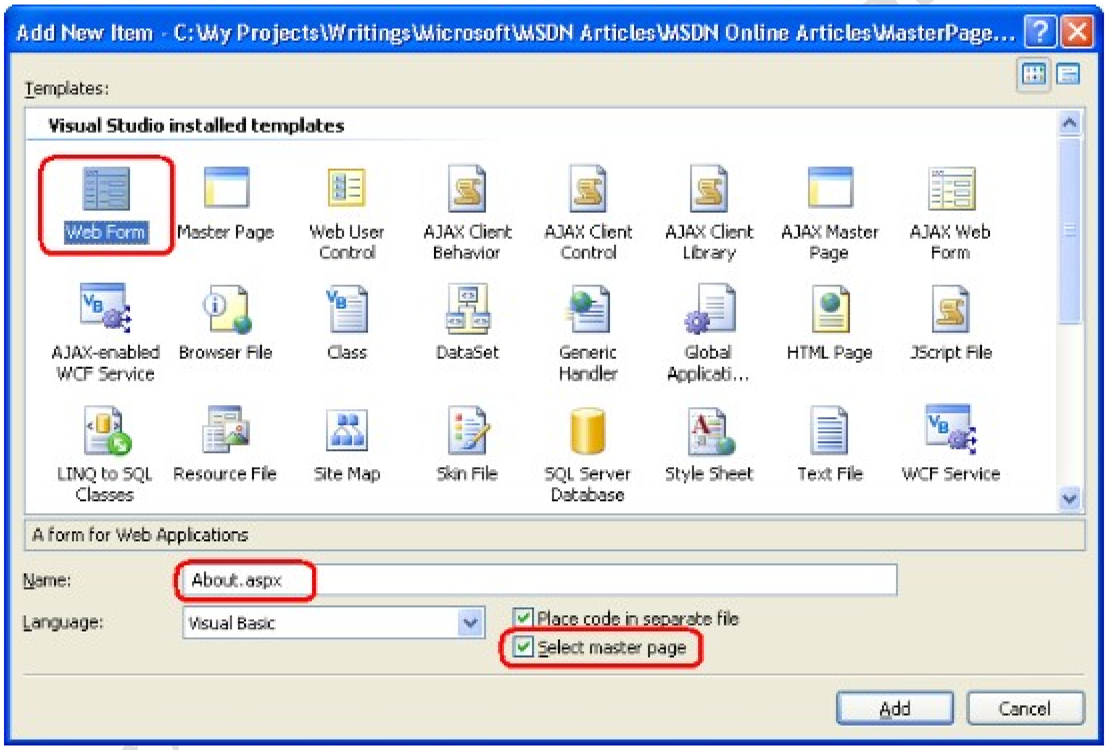
Fortunately, there are innumerous websites that offer free HTML design templates.One of my favorite ones is OpenDesigns.org. Once you find a website template you like, add the CSS files and images to your website project and integrate the template's HTML into your master page.

Note: Microsoft also offers a number of free ASP.NET Design Start Kit Templates that integrate into the New Web Site dialog box in Visual Studio.

Step 2: Creating Associated Content Pages

With the master page created, we are ready to start creating ASP.NET pages that are bound to the master page. Such pages are referred to as *content pages*.

Let's add a new ASP.NET page to the project and bind it to the Site.master master page. Right-click on the project name in Solution Explorer and choose the Add New Item option. Select the Web Form template, enter the name About.aspx, and then check the "Select master page" checkbox as shown in Figure below. Doing so will display the Select a Master Page dialog box (see Figure next) from where you can choose the master page to use.



Add New Item - C:\My Projects\Writings\Microsoft\MSDN Articles\MSDN Online Articles\MasterPage...

A form for Web Applications

Name: About.aspx

Language: Visual Basic

☑ Place code in separate file

☑ Select master page

Figure : Add a New Content Page

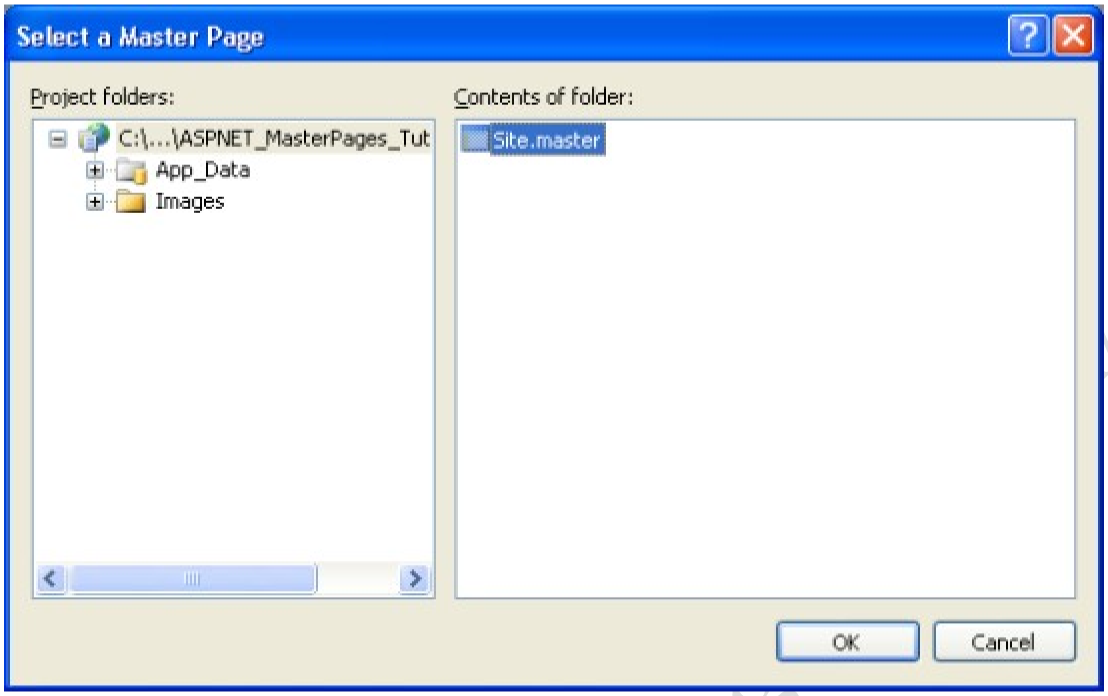


Figure : Select the Site.master Master Page

As the following declarative markup shows, a new content page contains a @Page directive that points back to its master page and a Content control for each of the master page's ContentPlaceHolder controls.

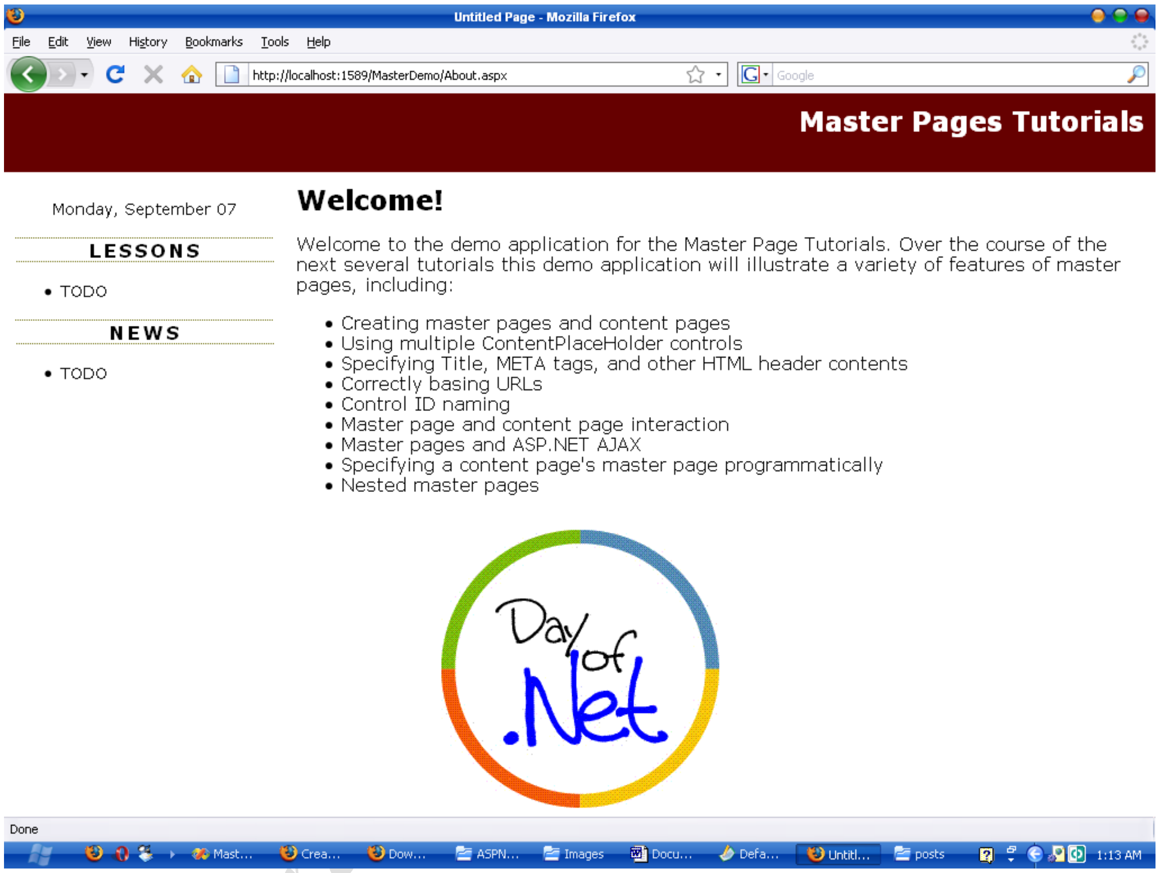
```
<%@ Page Language="VB" MasterPageFile="~/Site.master" AutoEventWireup="false"
CodeFile="About.aspx.vb" Inherits="About" Title="Untitled Page" %>

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

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

When rendering a content page, the ASP.NET engine must fuse the page's Content controls with its master page's ContentPlaceHolder controls. The ASP.NET engine determines the content page's master page from the @Page directive's MasterPageFile attribute. As the above markup shows, this content page is bound to ~/Site.master.

Because the master page has two ContentPlaceHolder controls - head and MainContent - Visual Studio 2005generated two Content controls. Each Content control references a particular ContentPlaceHolder via its ContentPlaceHolderID property.

Adding Markup and Web Controls to the Content Page

Take a moment to create some content for the About.aspx page. As you can see in Figure below, I entered an "Welcome" heading and a couple of paragraphs of text, but feel free to add Web controls, too. After creating this interface, visit the About.aspx page through a browser.


Untitled Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:1589/MasterDemo/About.aspx
Google
Done
Mast... Crea... Dow... ASPN... Images Docu... Defa... Untitl... posts
1:13 AM

Master Pages Tutorials

Monday, September 07

LESSONS

- TODO

NEWS

- TODO

Welcome!

Welcome to the demo application for the Master Page Tutorials. Over the course of the next several tutorials this demo application will illustrate a variety of features of master pages, including:

- Creating master pages and content pages
- Using multiple ContentPlaceHolder controls
- Specifying Title, META tags, and other HTML header contents
- Correctly basing URLs
- Control ID naming
- Master page and content page interaction
- Master pages and ASP.NET AJAX
- Specifying a content page's master page programmatically
- Nested master pages

Figure : Visit the About.aspx Page Through a Browser

It is important to understand that the requested content page and its associated master page are fused and rendered as a whole entirely on the web server. The end user's browser is then sent the resulting, fused HTML. To verify this, view the HTML received by the browser by going to the View menu and choosing Source. Note that there are no frames or any other specialized techniques for displaying two different web pages in a single window.

Binding a Master Page to an Existing ASP.NET Page

As we saw in this step, adding a new content page to an ASP.NET web application is as easy as checking the "Select master page" checkbox and picking the master page. Unfortunately, converting an existing ASP.NET page to a master page is not as easy.

To bind a master page to an existing ASP.NET page you need to perform the following steps:

1. Add the MasterPageFile attribute to the ASP.NET page's @Page directive, pointing it to the appropriate master page.
2. Add Content controls for each of the ContentPlaceHolders in the master page.
3. Selectively cut and paste the ASP.NET page's existing content into the appropriate Content controls. I say "selectively" here because the ASP.NET page likely contains markup that's already expressed by the master page, such as the DOCTYPE, the <html> element, and the Web Form.

Because it is much easier to create new content pages than it is to convert existing ASP.NET pages into content pages, I recommend that whenever you create a new ASP.NET website add a master page to the site. Bind all new ASP.NET pages to this master page. Don't worry if the initial master page is very simple or plain; you can update the master page later.

Note: When creating a new ASP.NET application, Visual Studio 2005adds a Default.aspx page that isn't bound to a master page. If you want to practice converting an existing ASP.NET page into a content page, go ahead and do so with Default.aspx. Alternatively, you can delete Default.aspx and then re-add it, but this time checking the "Select master page" checkbox.

Step 3: Updating the Master Page's Markup

One of the primary benefits of master pages is that a single master page may be used to define the overall layout for numerous pages on the site. Therefore, updating the site's look and feel requires updating a single file - the master page.

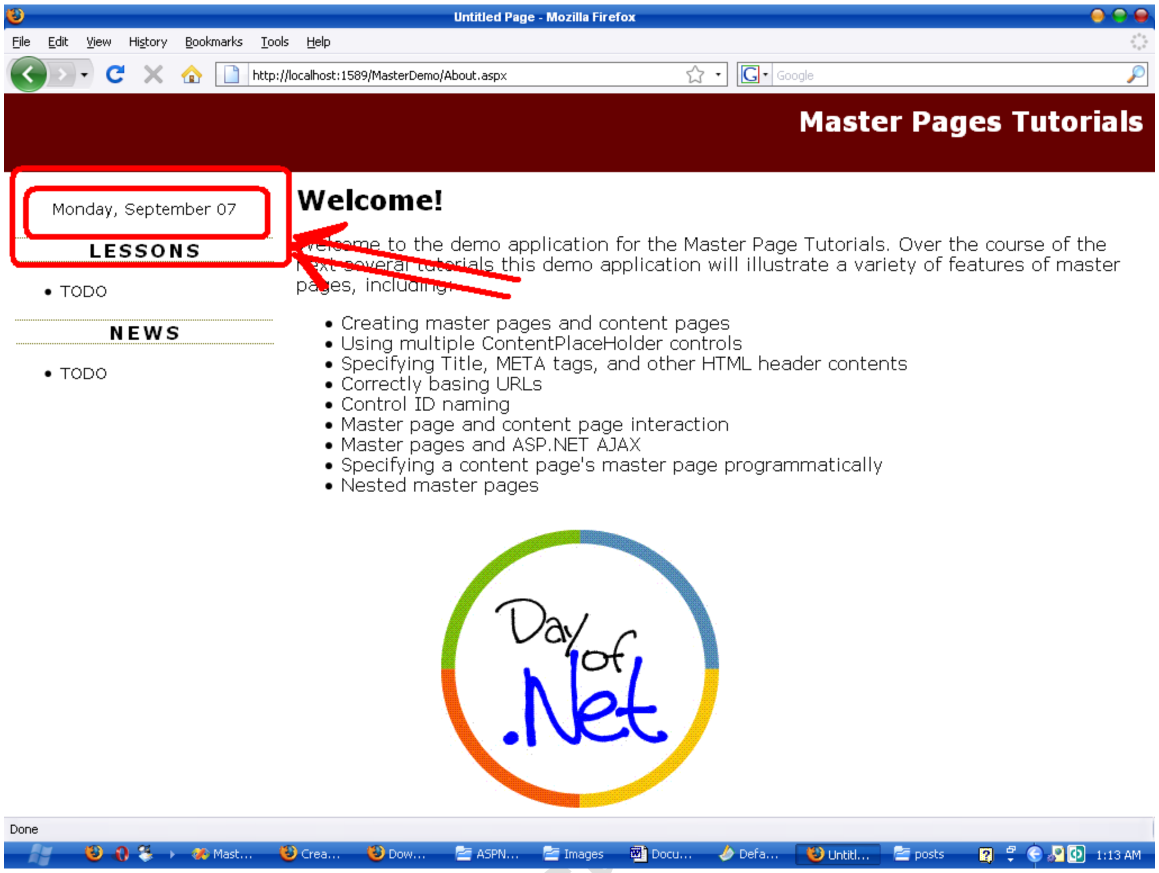
To illustrate this behavior, let's update our master page to include the current date in at the top of the left column. Add a Label named DateDisplay to the leftContent <div>.

```
<div id="leftContent">
<p style="text-align: center;">
<asp:Label ID="DateDisplay" runat="server">
</asp:Label>
</p>
<h3>Lessons</h3>
<ul>
<li>TODO</li>
</ul>
<h3>News</h3>
<ul>
<li>TODO</li>
</ul> </div>
```

Next, create a Page_Load event handler for the master page and add the following code:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load

DateDisplay.Text = DateTime.Now.ToString("dddd, MMMM dd") End Sub
```

The above code sets the Label's Text property to the current date and time formatted as the day of the week, the name of the month, and the two-digit day (see Figure). With this change, revisit one of your content pages. As Figure shows, the resulting markup is immediately updated to include the change to the master page.


Untitled Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:1589/MasterDemo/About.aspx
Google
Done

Master Pages Tutorials

Monday, September 07

LESSONS

- TODO

NEWS

- TODO

Welcome!

Welcome to the demo application for the Master Page Tutorials. Over the course of the next several tutorials this demo application will illustrate a variety of features of master pages, including:

- Creating master pages and content pages
- Using multiple ContentPlaceHolder controls
- Specifying Title, META tags, and other HTML header contents
- Correctly basing URLs
- Control ID naming
- Master page and content page interaction
- Master pages and ASP.NET AJAX
- Specifying a content page's master page programmatically
- Nested master pages

Figure : The Changes to the Master Page are Reflected When Viewing the a Content

Note: As this example illustrates, master pages may contain server-side Web controls, code, and event handlers.

Summary

Master pages enable ASP.NET developers to design a consistent site-wide layout that is easily updateable. Creating master pages and their associated content pages is as simple as creating standard ASP.NET pages, as Visual Studio 2005offers rich design-time support.

Happy Programming!