Themes and Master Pages

Today you will learn

- Styles
- Themes
- Master Pages

Styles

Style Types

Web pages can use styles in three different ways:

- *Inline style*: An inline style is a style that's placed directly inside an HTML tag. This can get messy, but it's a reasonable approach for one-time formatting. You can remove the style and put it in a style sheet later.
- *Internal style sheet*: An internal style sheet is a collection of styles that are placed in the <head> section of your web page markup. You can then use the styles from this style sheet to format the web controls on that page. By using an internal style sheet, you get a clear separation between formatting (your styles) and content (the rest of your HTML markup). You can also reuse the same style for multiple elements.
- *External style sheet*: An external style sheet is similar to an internal style sheet, except it's placed in a completely separate file. This is the most powerful approach because it gives you a way to apply the same style rules to many pages.

Creating a Basic Inline Style

To apply style to ordinary HTML element, you set the style attribute

```
<p style="background: Blue">This text has a blue background.</p>
```

While setting the multiple style properties, the properties are separated by semicolon.

```
<p style="color:White; background:Blue; font-size:x-large; padding:10px">
This text has a blue background.</p>
```

Creating a Basic Inline Style

Same approach can be used to apply formatting to web control (using style), but webcontrol provide formatting properties

```
<asp:Label ID="MyLabel" runat="server" ForeColor="White" BackColor="Blue"
 Font-Size="X-Large">Formatted Text</asp:Label>
```

It is actually rendered into following HTML, which uses an inline style

```
<span id="MyLabel" style="color:White; background-color:Blue; font-size:X-Large">
Formatted Text</span>
```

Style Builder

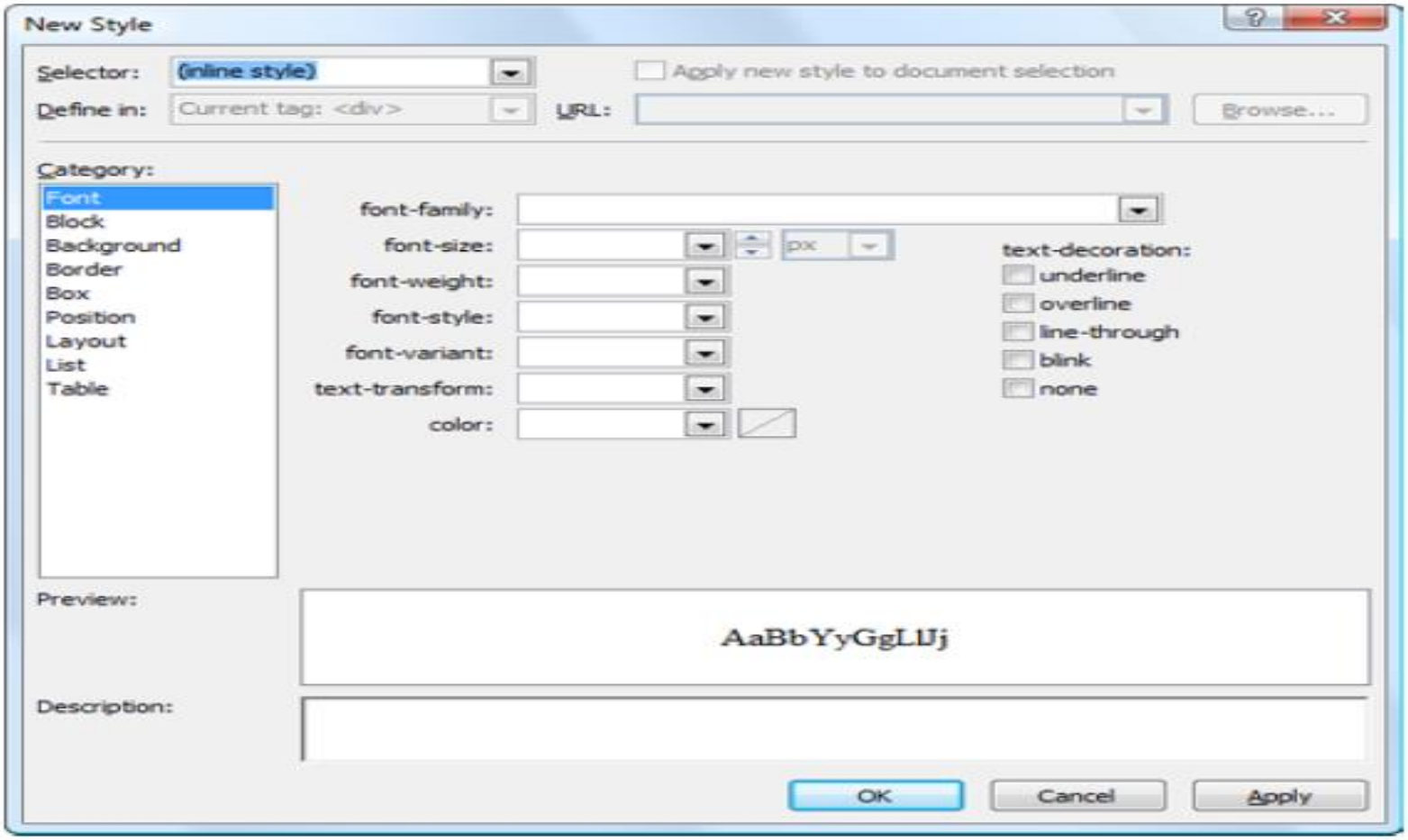
Style builder is used to create styles by picking and choosing your style preference in a dedicated style box

- Adding a style to a

```
<div>

    <asp:Label ID="Label1" runat="server">Type something here:
    </asp:Label>
    <asp:TextBox ID="TextBox1" runat="server">
    </asp:TextBox>
    <br /><br />
    <asp:Button ID="Button1" runat="server" Text="Button">
    </asp:Button>
</div>
```

Style Builder

- Creating an inline style



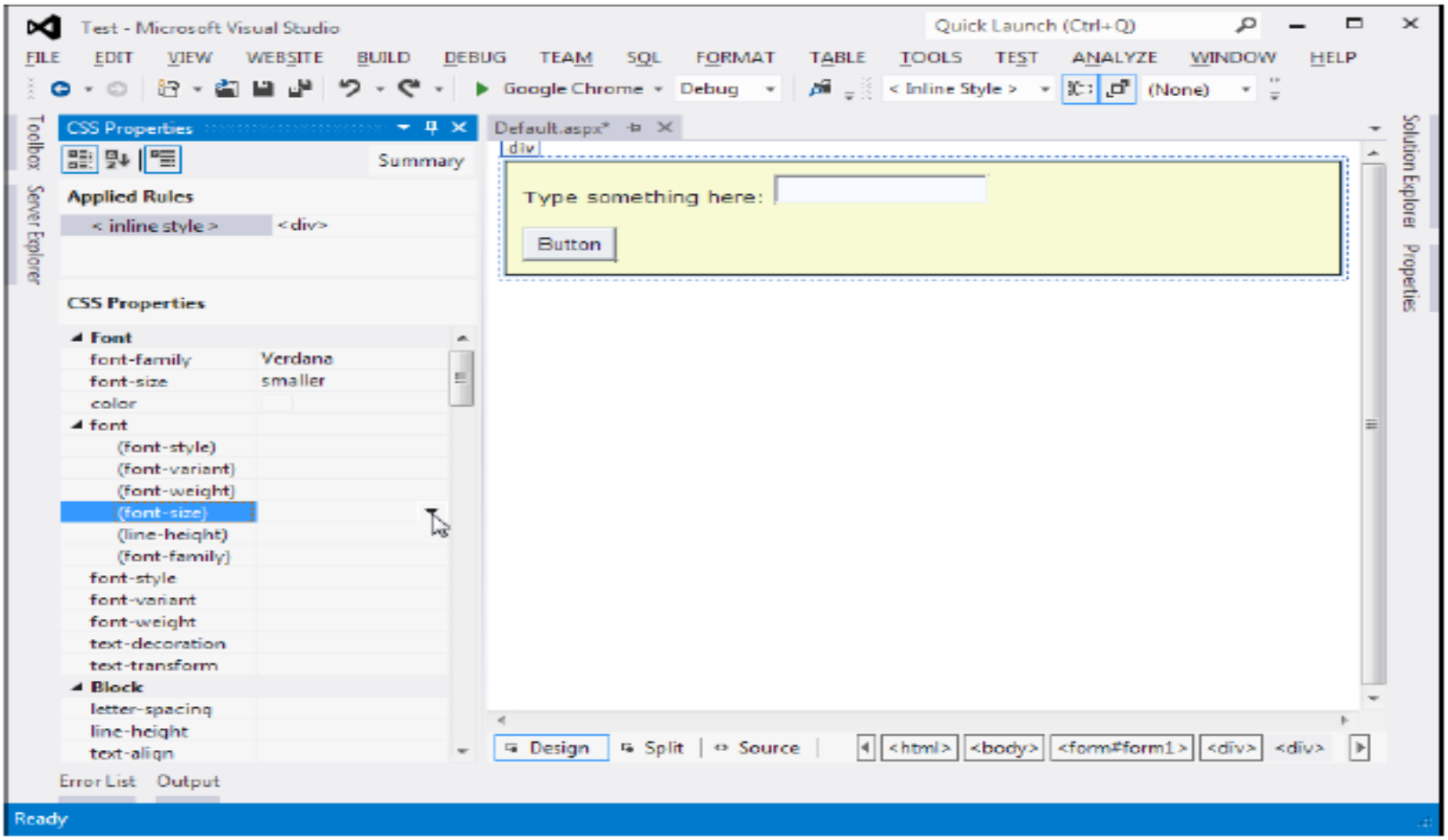
Style Builder

- Style setting in the New Style Dialog Box

Category	Description
Font	Allows you to choose the font family, font size, and text color, and apply other font characteristics (like italics and bold).
Block	Allows you to fine-tune additional text settings, such as the height of lines in a paragraph, the way text is aligned, the amount of indent in the first list, and the amount of spacing between letters and words.
Background	Allows you to set a background color or image.
Border	Allows you to define borders on one or more edges of the element. You can specify the border style, thickness, and color of each edge.
Box	Allows you to define the margin (the space between the edges of the element and its container) and the padding (the space between the edges of the element and its nested content inside).
Position	Allows you to set a fixed width and height for your element, and use absolute positioning to place your element at a specific position on the page. Use these settings with care. When you make your element a fixed size, there's a danger that the content inside can become too big (in which case it leaks out the bottom or the side). When you position your element using absolute coordinates, there's a chance that it can overlap another element.
Layout	Allows you to control miscellaneous layout settings. You can specify whether an element is visible or hidden, whether it floats at the side of the page, and what cursor appears when the user moves the mouse overtop, among other settings.
List	If you're configuring a list (a or element), you can set the numbering or bullet style. These settings aren't commonly used in ASP.NET web pages because you're more likely to use ASP.NET list controls like the BulletedList.
Table	Allows you to set details that only apply to table elements (such as <tr> and <td>). For example, you can control whether borders appear around an empty cell.

CSS Properties Window

- CSS Properties Window (View → CSS Properties)



Creating a Style Sheet

- Create a style sheet (Website → Add New Style Sheet template)
- In style sheet, several styles (also known as rules) can be define

```
.heading1
{
    font-weight: bold;
    font-size: large;
    color: limegreen;
    font-family: Verdana, Arial, Sans-Serif;
}
```

```
body
{
    font-family: Verdana, Arial, Sans-Serif;
    font-size: small;
}
```

- Each rule has two parts. The portion before the period indicates the HTML element to which the rule applies
- The portion after the period indicates the unique name (also called as CSS class name)

Applying Style Sheet Rules

- Link the page to appropriate style sheet. The <link> element references the file with the style sheet, which is placed in the <head> section of your page

```
<head runat="server">
  <title>...</title>
  <link href="StyleSheet.css" rel="stylesheet" />
</head>
```

Applying Style Sheet Rules

- To bind the style rules to the ASP.NET controls use CssClass Property

```
<asp:Label ID="Label1" runat="server" Text="This Label Uses heading1"
 CssClass="heading1"></asp:Label>
```

- To apply a style to an ordinary HTML element use Class property

```
<div class="blockText" id="paragraph" runat="server">
  <p>This paragraph uses the blockText style.</p>
</div>
```

Themes

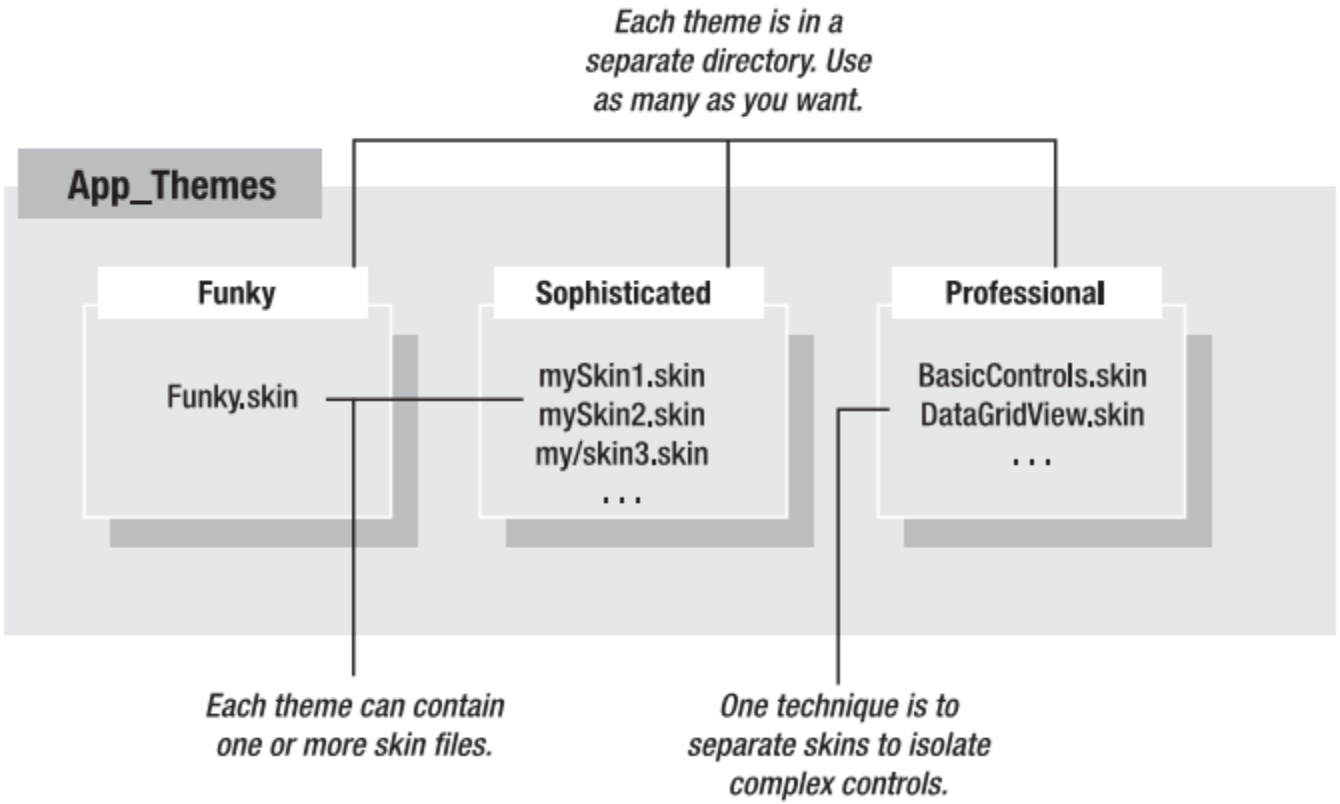
- To use theme in web application, you need to create a folder that defines it.
- The created folder should be placed in the App_Themes folder: placed inside the top-level directory for your web application.
- An application can contain definitions for multiple themes, but each themes should be in separate folder
- Only one theme can be active on a given page at a time.

Themes

- To make the theme accomplish the work, creation of at least one skin file in the theme folder is necessary
- A skin file is a text file with the .skin extension.
- Skin file is essentially a list of control tags, the control tags don't need to completely define the control
- Set the properties that need to be standardized

```
<asp:ListBox runat="server" ForeColor="White" BackColor="Orange"/>
```

Themes


Each theme is in a separate directory. Use as many as you want.
App_Themes
Funky
Funky.skin
Sophisticated
mySkin1.skin
mySkin2.skin
my/skin3.skin
. . .
Professional
BasicControls.skin
DataGridView.skin
. . .
Each theme can contain one or more skin files.
One technique is to separate skins to isolate complex controls.

Applying a Simple Theme

- To add a theme to your project, select Website ➤ Add New Item, and choose Skin File.


```
<asp:ListBox runat="server" ForeColor="White" BackColor="Orange"/>
<asp:TextBox runat="server" ForeColor="White" BackColor="Orange"/>
<asp:Button runat="server" ForeColor="White" BackColor="Orange"/>
```

```
<%@ Page Language="C#" AutoEventWireup="true" ... Theme="FunkyTheme" %>
```

Handling Theme Conflicts

- Conflict between your controls and your theme, the theme wins.
- To Override, use

```
<%@ Page Language="C#" AutoEventWireup="true" ... StyleSheetTheme="FunkyTheme" %>
```

- It's possible to use both the Theme attribute and the StyleSheetTheme
- Other approach,

```
<asp:Button ID="Button1" runat="server" ... EnableTheming="False" />
```

APPLYING A THEME TO AN ENTIRE WEBSITE

- Configure the <pages> element in the web.config file for your application

```
<configuration>
  <system.web>
    <pages theme="FunkyTheme">
      ...
    </pages>
  </system.web>
</configuration>
```

```
<configuration>
  <system.web>
    <pages styleSheetTheme="FunkyTheme">
      ...
    </pages>
  </system.web>
</configuration>
```

Creating Multiple Skins for the Same Control

- ASP.NET allows you to create multiple declarations for the same control
- Create a named skin by supplying a SkinID attribute

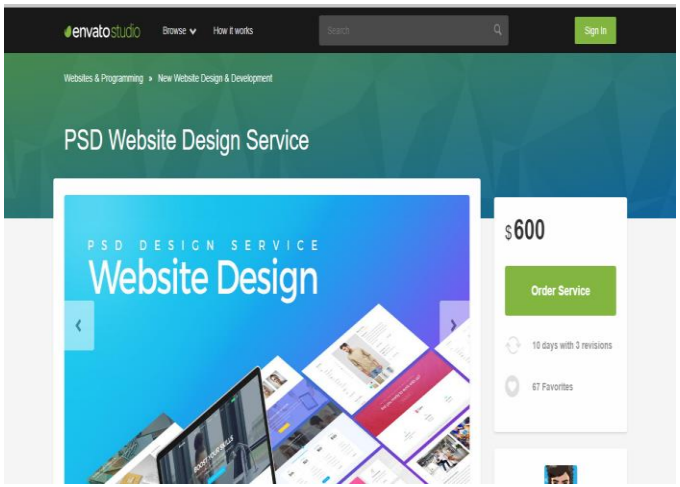
```
<asp:ListBox runat="server" ForeColor="White" BackColor="Orange" />
<asp:TextBox runat="server" ForeColor="White" BackColor="Orange" />
<asp:Button runat="server" ForeColor="White" BackColor="Orange" />

<asp:TextBox runat="server" ForeColor="White" BackColor="DarkOrange"
 Font-Bold="True" SkinID="Dramatic"/>
<asp:Button runat="server" ForeColor="White" BackColor="DarkOrange"
 Font-Bold="True" SkinID="Dramatic"/>
```

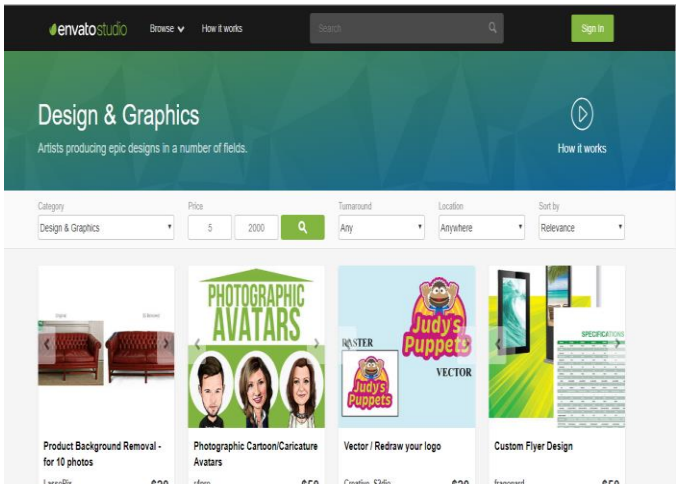
More Advanced Skins

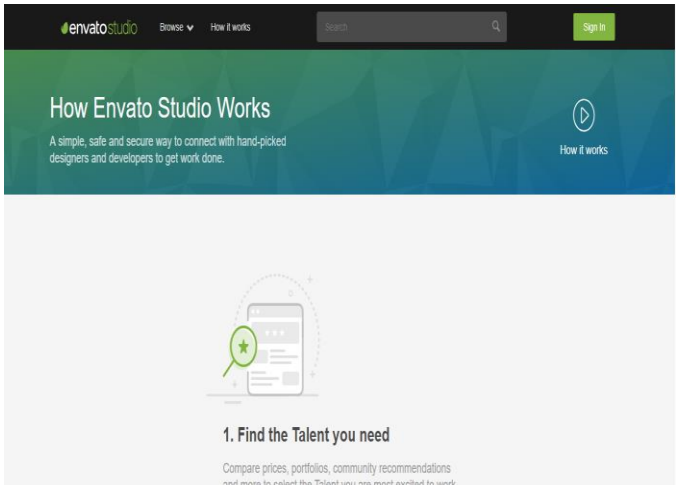
- Ex: Calendar Control

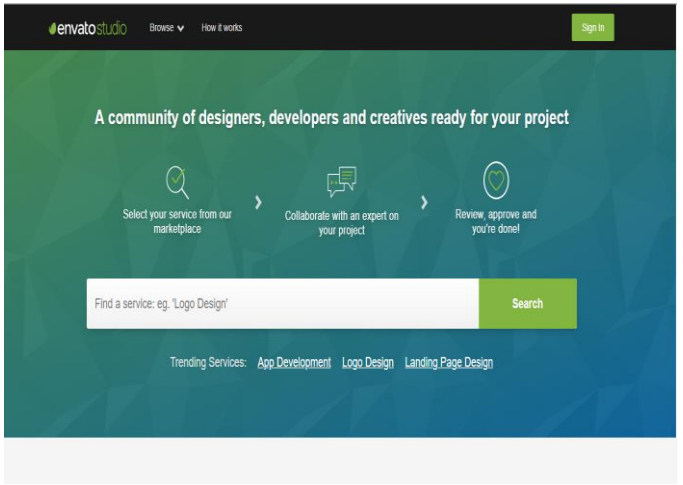
```
<asp:Calendar runat="server" BackColor="White" ForeColor="Black"
 BorderColor="Black" BorderStyle="Solid" CellSpacing="1"
 Font-Names="Verdana" Font-Size="9pt" Height="250px" Width="500px"
 NextPrevFormat="ShortMonth" SelectionMode="Day">
  <SelectedDayStyle BackColor="DarkOrange" ForeColor="White" />
  <DayStyle BackColor="Orange" Font-Bold="True" ForeColor="White" />
  <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White" />
  <DayHeaderStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333"
   Height="8pt" />
  <TitleStyle BackColor="Firebrick" BorderStyle="None" Font-Bold="True"
   Font-Size="12pt" ForeColor="White" Height="12pt" />
  <OtherMonthDayStyle BackColor="NavajoWhite" Font-Bold="False"
   ForeColor="DarkGray" />
</asp:Calendar>
```

Master Page


envatostudio
Browse
How it works
Search
Sign In
Websites & Programming » New Website Design & Development
PSD Website Design Service
PSD DESIGN SERVICE
Website Design
$600
Order Service
10 days with 3 revisions
67 Favorites


envatostudio
Browse
How it works
Search
Sign In
Design & Graphics
Artists producing epic designs in a number of fields.
How it works
Category
Price
Turnaround
Location
Sort by
Design & Graphics
5
2000
Any
Anywhere
Relevance
PHOTOGRAPHIC AVATARS
Judy's Puppets
VECTOR
Product Background Removal - for 10 photos
Photographic Cartoon/Caricature Avatars
Vector / Redraw your logo
Custom Flyer Design


envatostudio
Browse
How it works
Search
Sign In
How Envato Studio Works
A simple, safe and secure way to connect with hand-picked designers and developers to get work done.
How it works
1. Find the Talent you need
Compare prices, portfolios, community recommendations


envatostudio
Browse
How it works
Sign In
A community of designers, developers and creatives ready for your project
Select your service from our marketplace
Collaborate with an expert on your project
Review, approve and you're done!
Find a service: eg. 'Logo Design'
Search
Trending Services: App Development Logo Design Landing Page Design

Master Page Basics

- Look and feel of application
- Common layout
- One solution: Use frames
- Master pages are text files that can contain HTML, web controls, and code
- Extension: .master
- **Master pages** must be used by other pages, which are known as **content pages**.

A Simple Master Page and Content Page

- Blank page that includes a ContentPlaceHolder control.
- Everything else that's set in the master page is unchangeable in a content page.
- Demo
  - Create a master page
  - Add an image above the ContentPlaceHolder
  - Add a footer below ContentPlaceHolder
  - Add content page which uses master page

How Master Pages and Content Pages Are Connected

- Master Page Directive

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

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <img src="apress.jpg" /><br />
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
        <i>This is a simple footer.</i>
    </form>
</body>
</html>
```

How Master Pages and Content Pages Are Connected

- Linking your page to the master page by adding an attribute to the Page directive

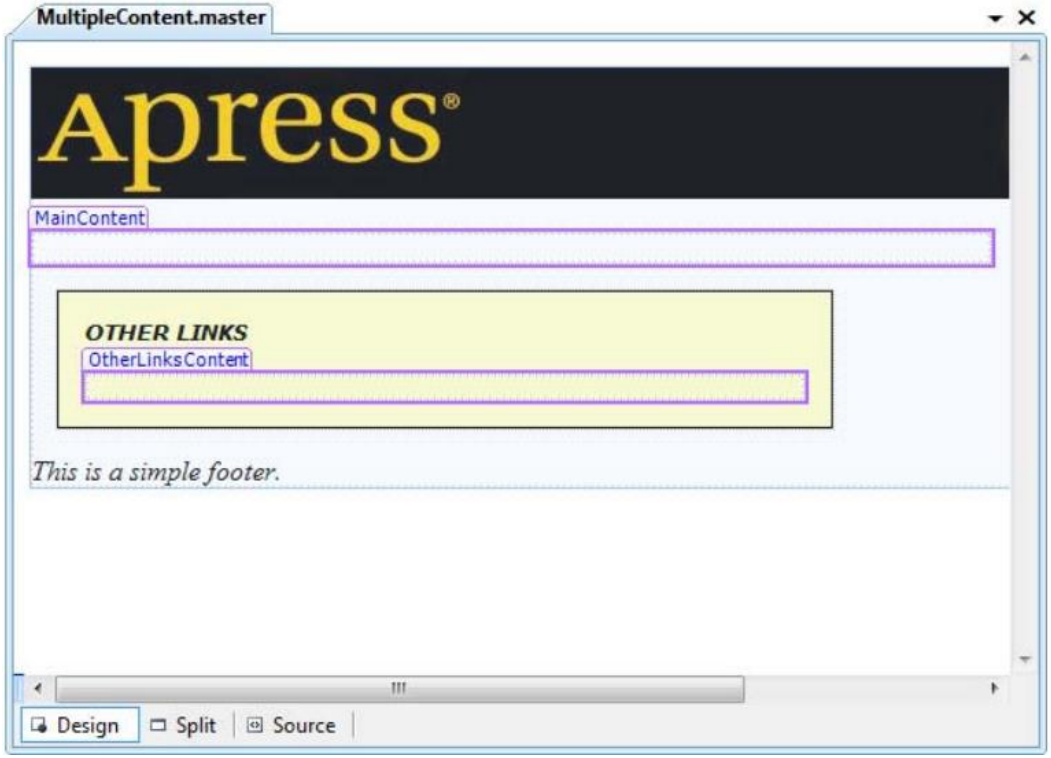
```
<%@ Page Language="C#" MasterPageFile="~/SiteTemplate.master"
    AutoEventWireup="true" CodeFile="SimpleContentPage.aspx.cs"
    Inherits="SimpleContentPage" Title="Untitled Page" %>
```

- Title attribute
- Place everything b/n <Content>

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
  runat="Server">
    <br />
    Here's some new content!
    <br />
</asp:Content>
```

A Master Page with Multiple Content Regions

- Add multiple ContentPlaceHolder controls and arrange them appropriately



A Master Page with Multiple Content Regions

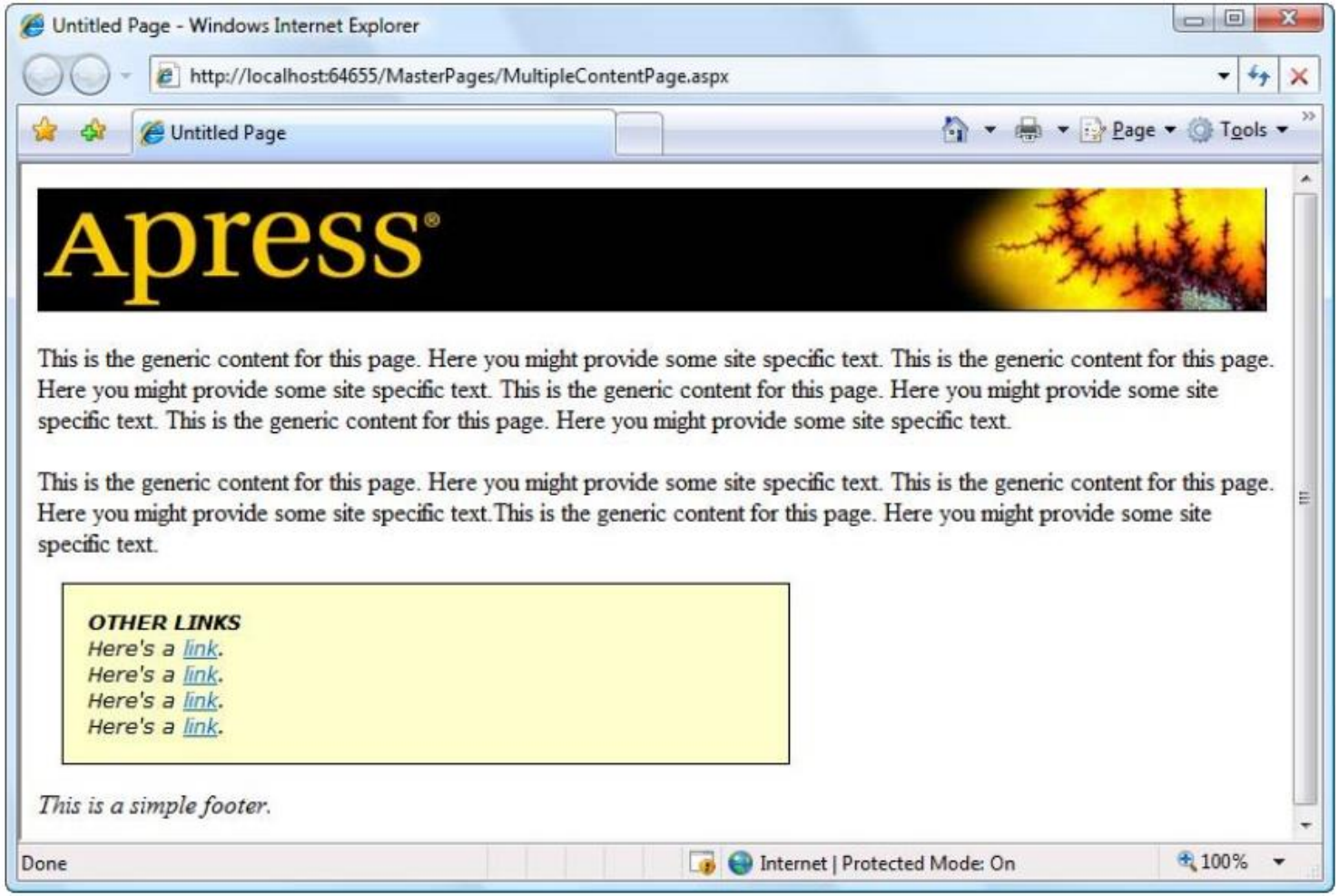
```
<body>
    <form id="form1" runat="server">
        <img src="apress.jpg" /><br />
        <asp:ContentPlaceHolder id="MainContent" runat="server">
        </asp:ContentPlaceHolder>
        <i>
            <div style="...">
                <b>OTHER LINKS</b>
                <br />
                <asp:ContentPlaceHolder id="OtherLinksContent" runat="server">
                </asp:ContentPlaceHolder>
            </div>
            This is a simple footer.
        </i>
```

A Master Page with Multiple Content Regions

- Content Page

```
<%@ Page Language="C#" MasterPageFile="~/MultipleContent.master"
    AutoEventWireup="true" CodeFile="MultipleContentPage.aspx.cs"
    Inherits="MultipleContentPage" Title="Content Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="Server">
    This is the generic content for this page. Here you might provide some site
    specific text ... </asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="OtherLinksContent"
  runat="Server">
    Here's a<a href="http://www.prosetech.com">link</a>.<br />
    ...
</asp:Content>
```

A Master Page with Multiple Content Regions



Default Content

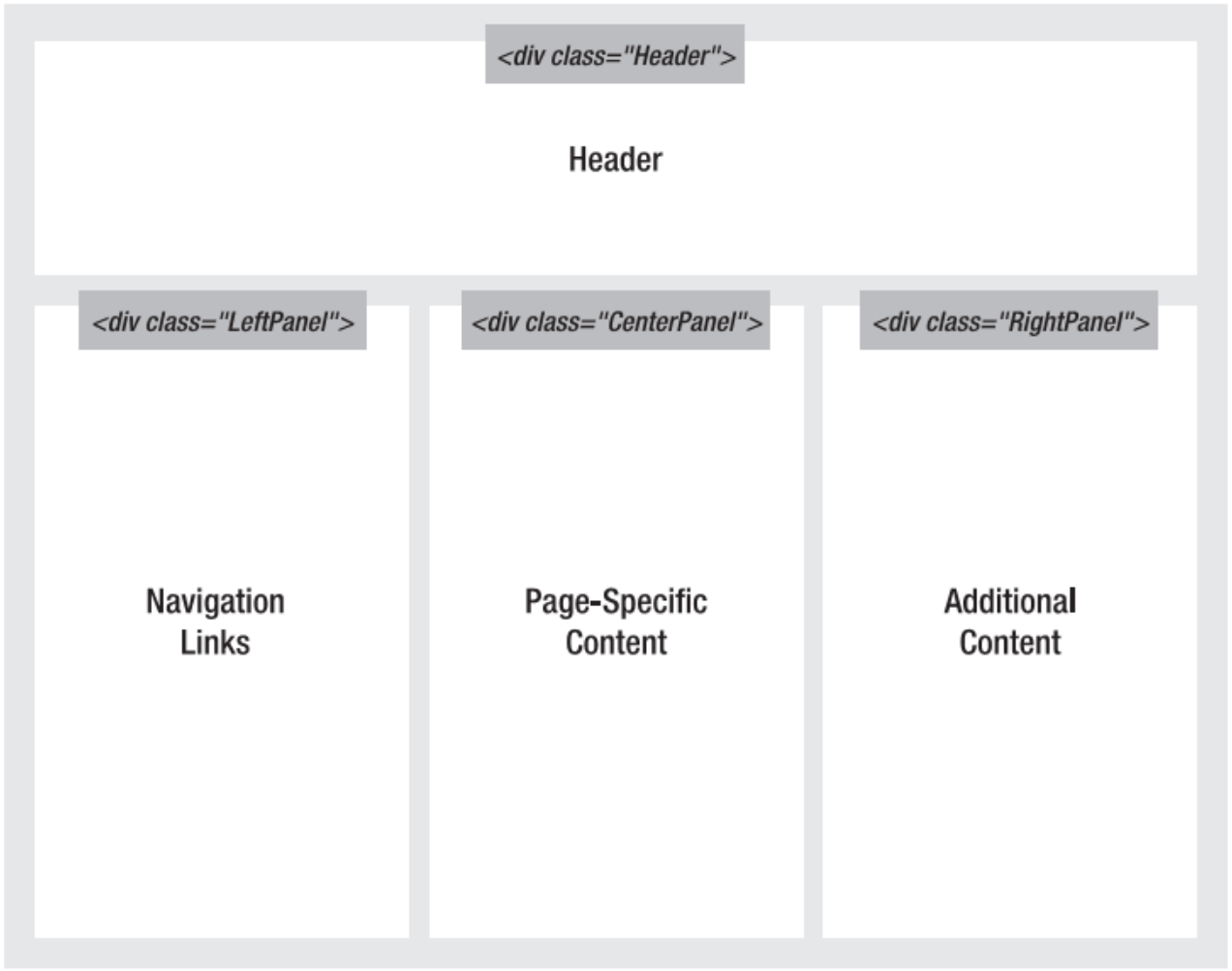
```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="SiteTemplate.master.cs"
    Inherits="SiteTemplate" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <img src="apress.jpg" /><br />
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
         This is default content.<br />
        </asp:ContentPlaceHolder>
        <i>This is a simple footer.</i>
    </form>
</body>
</html>
```

Style-Based Layouts

- More content is added, the page is reorganized and other content is bumped out of the way.

- To overcome this problem
  - HTML tables: Place ContentPlaceHolder in a single cell
  - CSS positioning: ContentPlaceHolder in each .

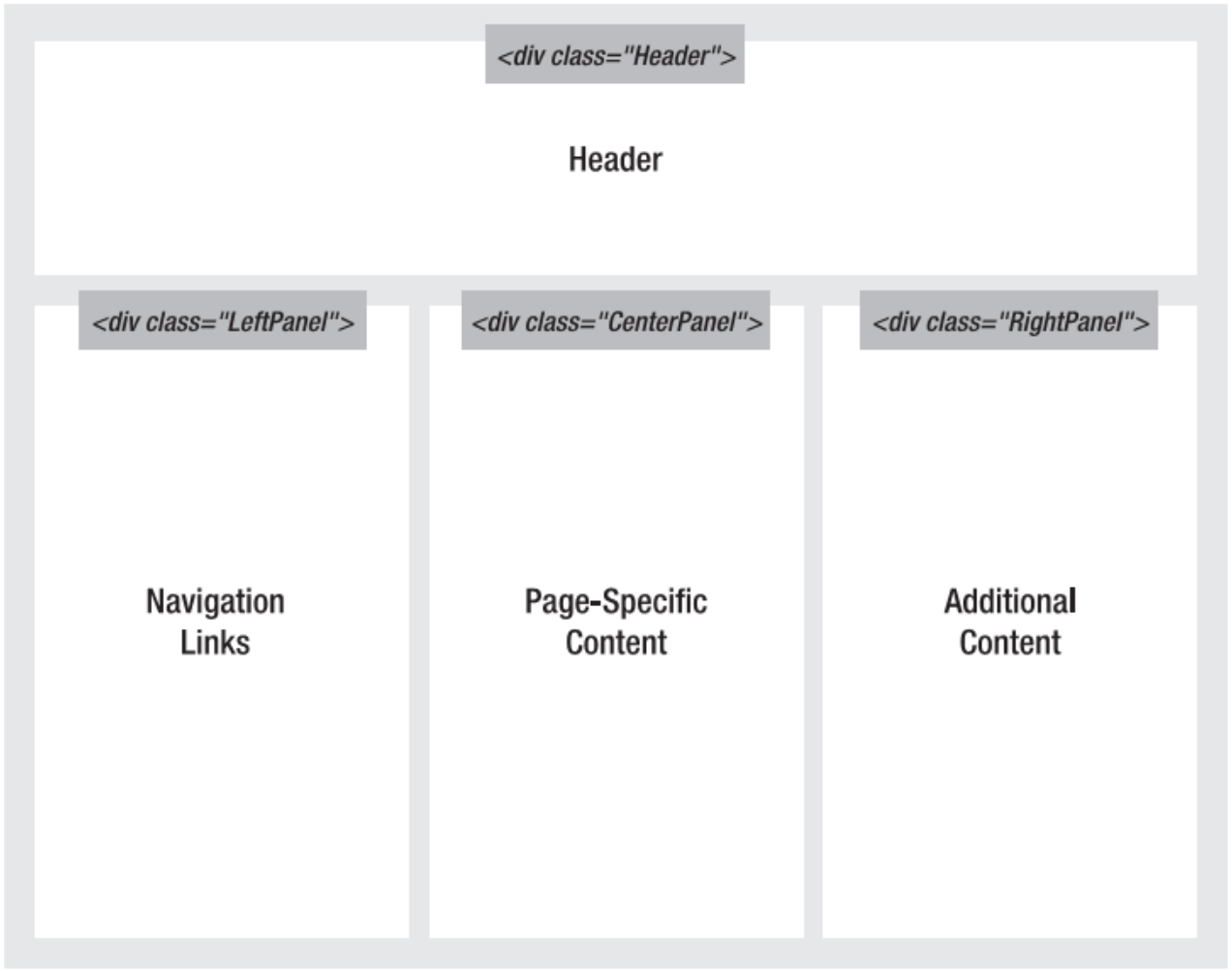
Style-Based Layouts




```
.Header
{
  position: absolute;
  top: 10px;
  left: 10px;
  height: 60px;
  text-align: center;
}
```

```
.LeftPanel
{
  position: absolute;
  top: 70px;
  left: 10px;
  width: 160px;
}
```

Style-Based Layouts

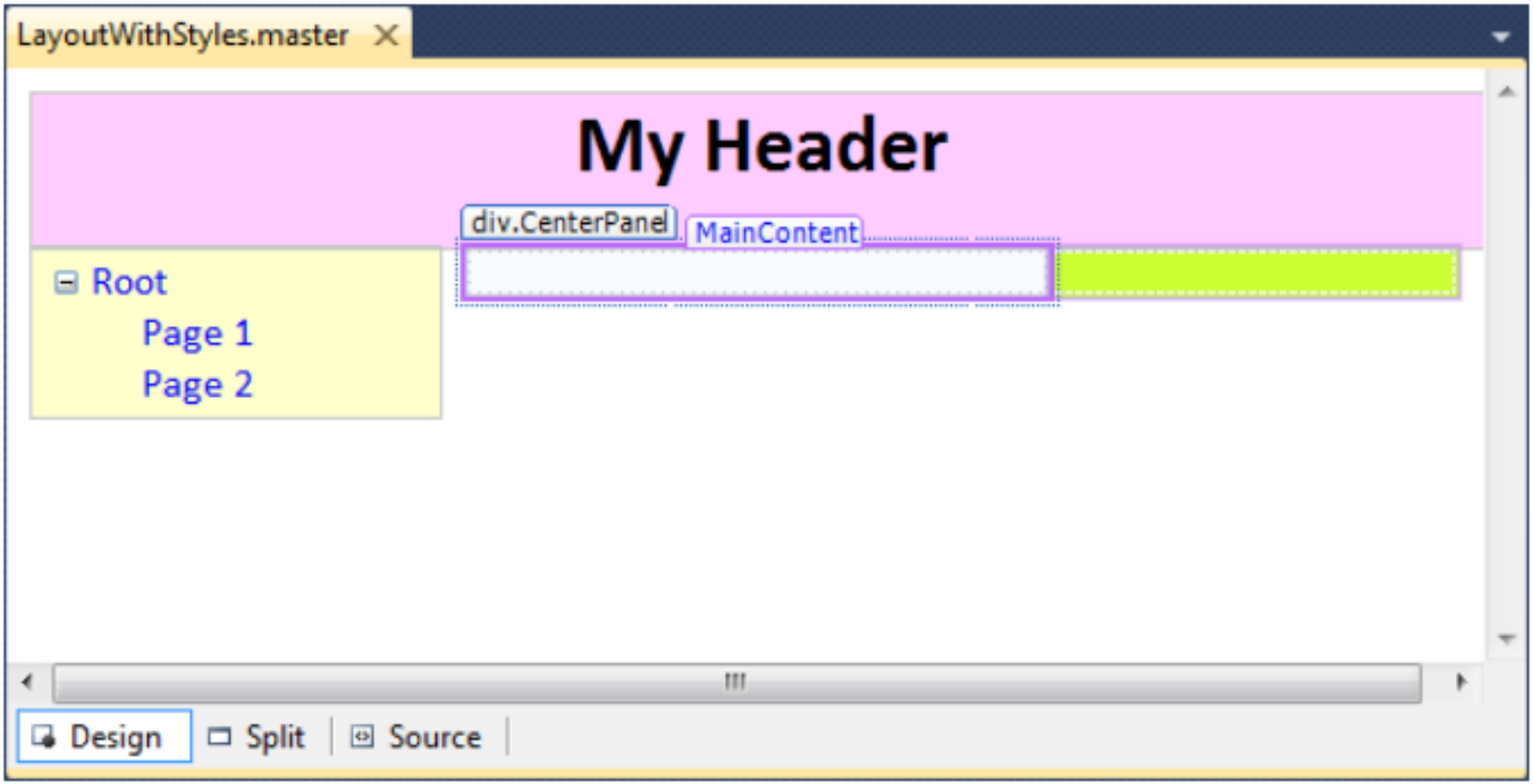



```
.RightPanel
{
  position: absolute;
  top: 70px;
  right: 10px;
  width: 160px;
}
```

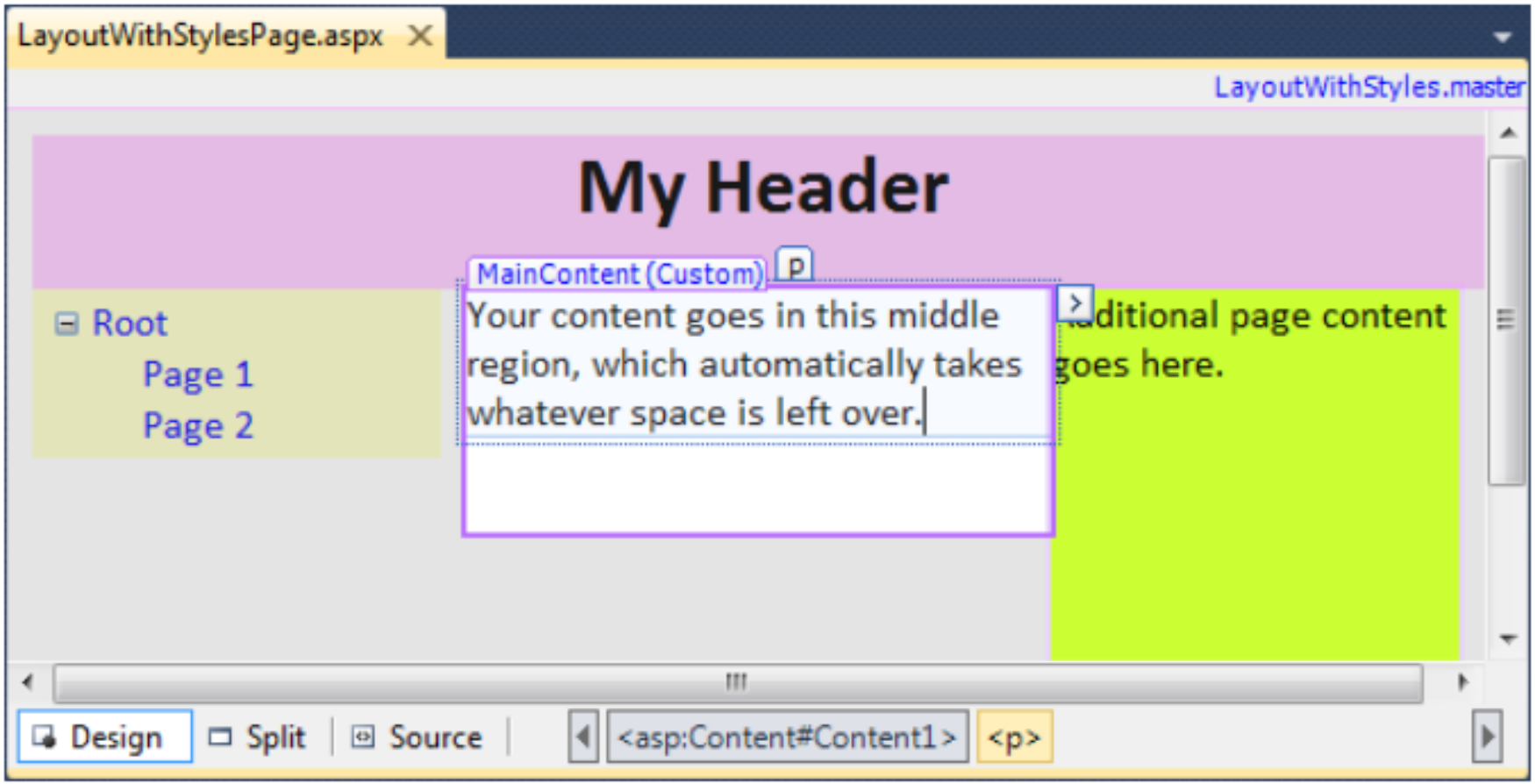
```
.CenterPanel
{
  position: absolute;
  top: 70px;
  margin-left: 175px;
  margin-right: 180px;
}
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="LayoutStyles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">

    <div class="Header">
      <h1>My Header</h1>
    </div>

    <div class="LeftPanel">
      <asp:TreeView ID="TreeView1" runat="server" Width="150px">
        <Nodes>
          <asp:TreeNode Text="Root" Value="New Node">
            <asp:TreeNode Text="Page 1" Value="Page 1"></asp:TreeNode>
            <asp:TreeNode Text="Page 2" Value="Page 2"></asp:TreeNode>
          </asp:TreeNode>
        </Nodes>
      </asp:TreeView>
    </div>
    <div class="CenterPanel">
      <asp:ContentPlaceHolder id="MainContent" runat="server">
      </asp:ContentPlaceHolder>
    </div>

    <div class="RightPanel">
      <asp:ContentPlaceHolder id="AdditionalContent" runat="server">
      </asp:ContentPlaceHolder>
    </div>
```

Style-Based Layouts



Style-Based Layouts



Interacting with a Master Page Programmatically

Interacting with a Master Page Programmatically

Interacting with a Master Page Programmatically

- Add a new property named ShowNavigationControls to the master page class.

```
public bool ShowNavigationControls
{
    get
    {
        return TreeVew1.Visible;
    }
    set
    {
        TreeView1.Visible = value;
    }
}
```

Interacting with a Master Page Programmatically

- Content page uses the built-in Page.Master property
- Cast the Page.Master object to the appropriate type

```
protected void cmdHide_Click(object sender, EventArgs e)
{
    TableMaster master = (TableMaster)this.Master;
    master.ShowNavigationControls = false;
}

protected void cmdShow_Click(object sender, EventArgs e)
{
    TableMaster master = (TableMaster)this.Master;
    master.ShowNavigationControls = true;
}
```

END OF LECTURE