# Master Pages

- A Master Page enables you to share the same content among multiple content pages in a website.
- Use a Master Page to create a common page layout.
- For example, if you want all the pages in your website to share a three column layout, you can create the layout once in a Master Page and apply the layout to multiple content pages.

# Creating Master Page

- You create a Master Page by creating a file that ends with the .master extension.

- Creating **SimpleMaster.master**

# Master Pages

```
<%@ Master Language=”C#” %>
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.1//EN”
“http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml” >
        <head id=”Head1” runat=”server”>
        <style type=”text/css”>
        html
        {
                background-color:silver;
                font:14px Arial,Sans-Serif;
        }
        .content
        {
                margin:auto;
                width:700px;
                background-color:white;
                border:Solid 1px black;
        }
```

# Master Pages

```
.leftColumn
{
        float:left;
        padding:5px;
        width:200px;
        border-right:Solid 1px black;
        height:700px;
}
.rightColumn
{
        float:left;
        padding:5px;
}
.clear
{
        clear:both;
}
```

# Master Pages

```
</style>
<title>Simple Master</title>
</head>
<body>
        <form id="form1" runat="server">
                <div class="content">
                        <div class="leftColumn">
                                <asp:contentplaceholder
                        id="ContentPlaceHolder1" runat="server"/>
                        </div>
```

# Master Pages

```
        <div class=”rightColumn”>
                <asp:contentplaceholder
                id=”ContentPlaceHolder2” runat=”server”/>
        </div>
        <br class=”clear” />
</div>
</form>
</body>
</html>
```

# Master Pages

Two special things about master pages are:

- The file contains a <%@ Master %> directive instead of the normal <%@ Page %> directive.
- Second, the Master Page includes two ContentPlaceHolder controls.
- You can add as many ContentPlaceHolders to a Master Page as you need.

# Using Master Page in .aspx page

**SimplePage.aspx**

```
<%@ Page Language=”C#” MasterPageFile=”~/SimpleMaster.master” %>
<asp:Content  ID=”Content1” ContentPlaceHolderID=”ContentPlaceHolder1” Runat=”Server”>
        Content in the first column
        <br />Content in the first column
        <br />Content in the first column
        <br />Content in the first column
        <br />Content in the first column
</asp:Content>
<asp:Content ID=”Content2” ContentPlaceHolderID=”ContentPlaceHolder2” Runat=”Server”>
        Content in the second column
        <br />Content in the second column
        <br />Content in the second column
        <br />Content in the second column
        <br />Content in the second column
</asp:Content>
```

# Master Pages

- The Master Page is associated with the content page through the MasterPageFile attribute included in the <%@ Page %> directive.
- This attribute contains the virtual path to a Master Page.
- All the content contained in the content page must be added with Content controls.
- The Content control includes a ContentPlaceHolderID property. This property points to the ID of a ContentPlaceHolder control contained in the Master Page.

# Master Pages

- Displaying default content using Master Pages.

# Master Pages

- Displaying default content using Master Pages.
- Nesting of Master Pages.

# Themes and Skin files

- An ASP.NET Theme enables you to apply a consistent style to the pages in your website.
- You can use a Theme to control the appearance of both the HTML elements and ASP.NET controls that appear in a page.
- Themes are different than Master Pages. A Master Page enables you to share content across multiple pages in a website. A Theme, on the other hand, enables you to control the appearance of the content.

# Creating Themes and Skins

- You create a Theme by adding a new folder to a special folder named ***App_Themes*** in your application.
- Each folder that you add to the App_Themes folder represents a different Theme.
- If the App_Themes folder doesn't exist in your application, you can create it. It must be located in the root of your application.

- The most important types of files in a Theme folder:
  - Skin Files
  - Cascading Style Sheet Files

# Adding Skins to Themes

- A Theme can contain one or more Skin files.
- A Skin enables you to modify any of the properties of an ASP.NET control that have an effect on its appearance.
- For example, imagine that you decide that you want every TextBox control in your web application to appear with a yellow background color and a dotted border.

- For this let us create a theme folder SimpleTheme in AppThemes Folder.
- And to modify the appearance of all TextBox controls in all pages that use the Simple Theme create a skin file TextBox.skin as shown below.
- (You can name a skin file anything that you want)

# Adding Skins

SimpleTheme \TextBox.skin

```
<asp:TextBox   BackColor=”Yellow”   BorderStyle=”Dotted”
        Runat=”Server” />
```

A Theme folder can contain Skin file that contains Skins for hundreds of controls.

A single Skin file can contain Skins for hundreds of controls.

To apply the theme to a particular page use Theme Property of Page Directive :

```
<%@ Page Language=”C#” Theme=”SimpleTheme” %>
```

# Creating Named Skins

- In the previous example, we created something called a Default Skin. A Default Skin is applied to every instance of a control of a certain type.
- When you create a Named Skin, you can decide when you want to apply the Skin.
- To create a named skin you define a ***SkinId*** in the skin file.
- A Theme can contain only one Default Skin for each type of control. However, a Theme can contain as many Named Skins as you want.
- Each Named Skin must have a unique name.

Example:

Here in the next example we have both default and named skins:

# Themes and Skin files

```
<asp:TextBox   SkinID=”DashedTextBox” BorderStyle=”Dashed”
BorderWidth=”5px” Runat=”Server” />

<asp:TextBox   BorderStyle=”Double”  BorderWidth=”5px”
Runat=”Server” />
```

# Applying named skin to the .aspx page

***ShowNamedSkin.aspx***

```
<%@ Page Language=”C#” Theme=”SimpleTheme” %>
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.1//EN”
“http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml” >
<head runat=”server”>
<title>Show Named Skin</title>
</head>
```

# Themes and Skin files

```
<body>
<form id=”form1” runat=”server”>
        <div>
        <asp:TextBox   id=”txtFirstName”  SkinID=”DashedTextBox”
        Runat=”server” />
                <br /><br />
        <asp:TextBox   id=”txtLastName”  Runat=”server” />
        </div>
</form>
</body>
</html>
```

# Themes and Skin files

- The above page contains two TextBox controls. The first TextBox control includes a SkinID attribute. This attribute causes the Named Skin to be applied to the control.
- The second TextBox, on the other hand, does not include a SkinID property. The Default Skin is applied to the second TextBox control.

# Thanks