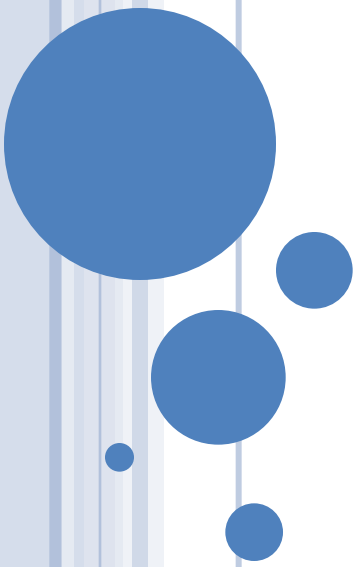
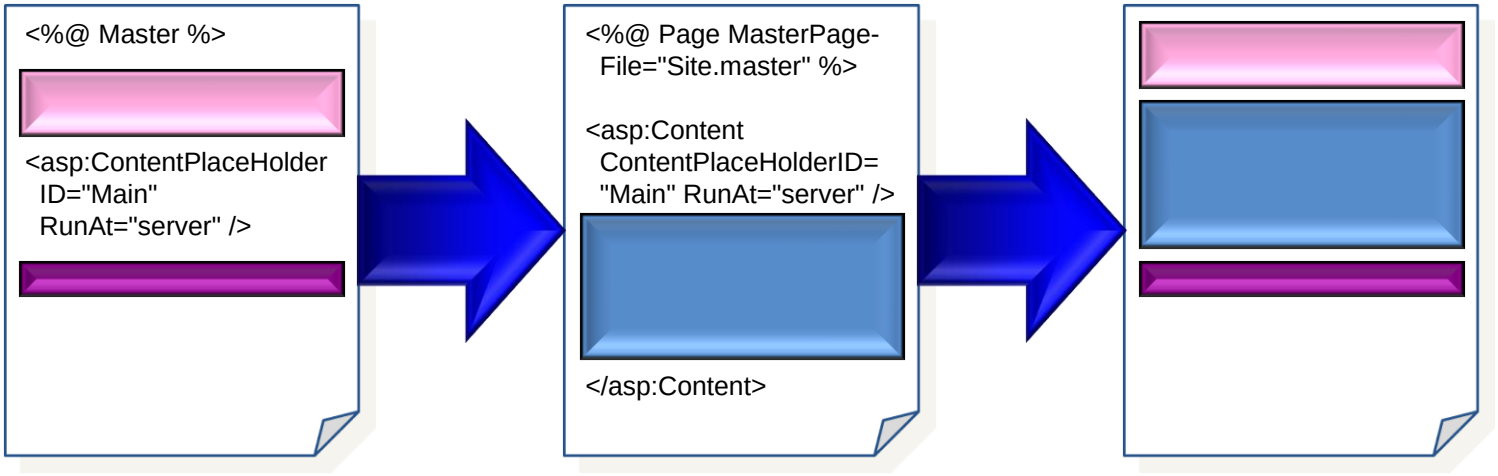


Master Pages & Themes

Master Page Basics



Creating a Master Page

```
<%@ Master %>
<html>
  <body>
    <!-- Banner shown on all pages that use this master -->
    <table width="100%">
      <tr>
        <td bgcolor="darkblue" align="center">
          <span style="font-size: 36pt; color: white">iConnect Software India</span>
        </td>
      </tr>
    </table>

    <!-- Placeholder for content below banner -->
    <asp:ContentPlaceHolder ID="Main" RunAt="server" />
  </body>
</html>
```

Applying Master Page

```
<%@ Page MasterPageFile="~/Site.master" %>

<asp:Content ContentPlaceHolderID="Main" RunAt="server">
  This content fills the place holder "Main" defined in the master page
</asp:Content>
```

```
<configuration>
  <system.web>
    <pages masterPageFile="~/Site.master" />
  </system.web>
</configuration>
```

Programmatically Applying Master

```
void Page_PreInit (Object sender, EventArgs e)
{
    Page.MasterPageFile = "~/Site.master";
}
```

- Each page has got property *Master* which exposes the various master page properties. (e.g. Title)

Accessing a Control (in Master) via Content Page

- Site.master

```
<asp:Label ID="Title" RunAt="server" />
```

- Content Page

```
( Master.FindControl ("Title") as Text).Text = "Orders";
```

@MasterType Directive

- By adding the *@MasterType* directive in the content page, one can avoid casting.

```
<%@ Page Language="C#" MasterPageFile="SimpleWithProp.master“
    CodeFile="HelloMasterType.aspx.cs" Inherits="HelloMasterType" %>

<%@ MasterType VirtualPath="SimpleWithProp.master" %>
```

- *VirtualPath* & *Typename* attribute
  - *Both serve to identify the master class to use. The former does it by URL; the latter does it by type name.*

Accessing a Control (in Master) via Content Page

- Site.master

```
<asp:Label ID="Title" RunAt="server" />
  …
<script language="C#" runat="server">
public string TitleText
{
    get { return Title.Text; }
    set { Title.Text = value; }
}
</script>
```

- Content Page

```
Master.TitleText = "Orders";
```

Nesting Master Pages

```
<!-- Orders.Master -->
<%@ Master MasterPageFile="~/Site.Master" %>

<asp:Content ContentPlaceHolderID="..." RunAt="server">
  <asp:ContentPlaceHolder ID="..." RunAt="server">
    ...
  </asp:ContentPlaceHolder>
```

Device-Specific Masters

```
<%@ Page masterpagefile="Base.master“
         ie:masterpagefile="ieBase.master“
          netscape6to9:masterpagefile="nsBase.master" %>
```

Order of Events

- **Master page child controls initialization.**
- **Content page child controls initialization.**
- **Master page initialization.**
- **Content page initialization.**

Order of Events

- **Content page load.**
- **Master page load.**
- **Master page child controls load.**
- **Content page child controls load.**

Caching

- To work with output caching when using a master page, stick the OutputCache directive in the content page.

Themes

- Themes maintain the style information separately from the page.
- Modifying a theme change the look of pages with that theme.
- Style sheets apply formatting to HTML elements, themes configure ASP.NET controls.

Themes

- All themes are application-specific. Hence created in application level folder *App_Themes*.

- Global Themes can be put in c:\Inetpub\wwwroot\aspnet_client\system_web\[Version]\Themes folder.

Applying a Theme

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

Applies to site

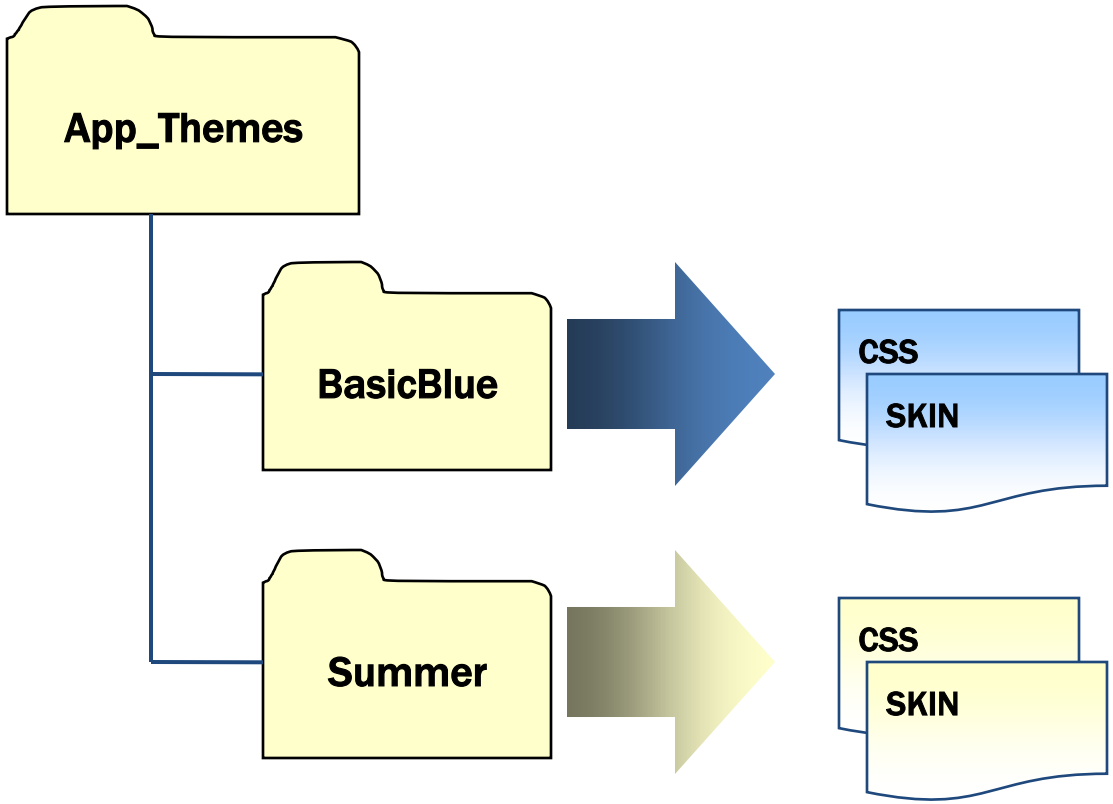
Applies to page

```
<%@ Page Theme="BasicBlue" %>
```

```
<script runat="server">
protected void Page_PreInit(object sender, System.EventArgs e)
{
Page.Theme ="BasicBlue";
}
</script>
```

Programmatically

Folder Structure



Theme

- Each theme folder must contain the elements of the theme.
- Theme can include
  - A single skin file
  - CSS files
  - Images

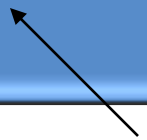
Creating a skin

- A *skin* is a definition of styles applied to the server controls. Skin file (a text file) has a extension ".skin".
- Skin sets only the properties that one want to standardize.
- In a skin file, for every control *runat="server"* is compulsory.
- Skin file do not include ID attribute for controls.

Creating a skin

```
<!-- Default look for DropDownList controls -->
<asp:DropDownList runat="server" BackColor="hotpink“
 ForeColor="white" />

<!-- Default look for DataGrid controls -->
<asp:DataGrid runat="server" BackColor="#CCCCCC" BorderWidth="2pt"
  BorderStyle="Solid" BorderColor="#CCCCCC" GridLines="Vertical"
  HorizontalAlign="Left">
  <HeaderStyle ForeColor="white" BackColor="hotpink" />
  <ItemStyle ForeColor="black" BackColor="white" />
  <AlternatingItemStyle BackColor="pink" ForeColor="black" />
</asp:DataGrid>
  ...
```



Sample of **some.skin** file

Skins with IDs

- Single skin file of theme lets us define multiple versions for single type of control.
- To create multiple style definitions of a single control, use *SkinID* attribute.
- The default skin is the one in the .skin file that doesn't have a SkinID attribute.

Skins With ID

```
<asp:Textbox Runat="server" ForeColor="#004000" Font-Names="Verdana"
Font-Size="X-Small" BorderStyle="Solid" BorderWidth="1px"
BorderColor="#004000" Font-Bold="True" />

<asp:Textbox Runat="server" ForeColor="#000000" Font-Names="Arial"
Font-Size="X-Large" BorderStyle="Dashed" BorderWidth="3px"
BorderColor="#000000" Font-Bold="False" SkinID="TextboxDashed" />
```

```
<asp:Textbox ID="TextBox1" Runat="server">Textbox1</asp:Textbox>

<asp:Textbox ID="TextBox2" Runat="server"
SkinId="TextboxDashed">Textbox2</asp:Textbox>
```

```
TextBox1.SkinID = "TextboxDashed";  //Programmatically
```

EnableTheming Attribute

- *EnableTheming* attribute if set to false, no theme can affect control appearance.

```
<asp:Textbox ID=”TextBox1” Runat=”server”
BackColor=”#000000” ForeColor=”#ffffff” EnableTheming=”false” />
```

```
<%@ Page Language=”C#” EnableTheming=”false” %>
```

StyleSheetTheme Attribute

- Just like *Theme* attribute.
- Difference is that local attribute settings override theme settings.

```
<%@ Page Language="C#" StylesheetTheme="Summer" %>
```

Web.config file

```
<system.web>
        <pages styleSheetTheme="Summer" />
</system.web>
```

Images in Theme

- Themes enable one to incorporate actual images into the style definitions.

- Some of the controls use images to create a better visual appearance.

e.g. Organize button images in *ButtonImages* folder in the theme folder.

```
<asp:ImageButton runat="server" skinID="OKButton
ImageUrl="ButtonImages/buttonOK.jpg" />
```