Agenda

About Master Page

- Master pages allow you to create a consistent look and behavior for all the pages (or group of pages) in your web application
- The extension of MasterPage is "**.master**"
- MasterPage cannot be directly accessed from the client because it just acts as a template for the other Content Pages.
- In a MasterPage we can have content either inside **ContentPlaceHolder** or outside it. Only content inside the **ContentPlaceholder** can be customized in the Content Page.
- We can have multiple masters in one web application.
- A MasterPage can have another MasterPage as Master to it.
- The content page content can be placed only inside the content tag.
- Controls of MasterPage can be programmed in the MasterPage itself and content page. A control in content page will never be programmed in MasterPage.
- A master page of one web application cannot be used in another web application.
- The **MasterPageFile** property of a webform can be set dynamically and it should be done either in or before the **Page_PreInit** event of the WebForm. **Page.MasterPageFile** = "MasterPage.master". The dynamically set Master Page must have the ContentPlaceHolder whose content has been customized in the WebForm.
- The order in which events are raised: **Load (Page)** → **Load (Master)** → **LoadComplete (Page)** i.e. if we want to overwrite something already done in Load event handler of Master then it should be coded in the LoadComplete event of the page.

Things to Understand:

- Significance of **ContentPlaceHolder** Tag in MasterPage and **Content** Tag in WebForm.
- How a control of MasterPage can be accessed / programmed in WebForm.
  - **Master.FindControl**
  - Public property in MasterPage and **<%@MasterType** directive in WebForm.

Example to demonstrate:

1. Add a new MasterPage file (MainMaster.master) to the Web Application.
2. Change the Id of ContentPlaceHolder in <Head> to "cphHead" and the Id "ContentPlaceHolder1" to "cphFirst"
3. Add one more ContentPlaceHolder (cphSecond) to Master page.
4. To the master page add some header, footer and some default content for both the content place holders.

Using ContentPlaceHolder

```
<form id="form1" runat="server">
Header...<br />
<asp:ContentPlaceHolder ID="cphFirst" runat="server">
  This is First Content Place Holder (Default) </asp: ContentPlaceHolder>
  <br />
```

```
        <asp:ContentPlaceHolder ID="cphSecond" runat="server">
            This is Second Content Place Holder (Default)
        </asp:ContentPlaceHolder>
        <br />
    Footer...
    </form>
```

5. To the web application add a WebForm (Default.aspx) → Check (Select Master Page) in New Item Dialog
6. Note the attribute "MasterPageFile" in @Page directive of the WebForm.
7. Delete the <content tag for the ContentPlaceHolderId="cphSecond".
8. Run the WebForm – The output rendered includes the Header, Footer, Contentof cphSecond in Master and the content of <content tag for ContentPlaceHolderId="cphFirst" in webform.
9. Here we understood the importance of ContentPlaceHolder in Master and Content in WebForm.
10. Add a Label in the master page (outside ContentPlaceHolder)

    `<asp:Label ID="lblMaster" runat="server" Text="In Master"/>`
11. Add a button to WebForm(inside content tag)

    `<asp:Button ID="btnDemo" runat="server" onclick="btnDemo_Click" Text="Set Label of Master" />`
12. Handle the Click event of above button and add to it the code as mentioned below.

btnDemo_Click event

```
protected void btnDemo_Click(object sender, EventArgs e)
{
    //Get reference to Label control (lblMaster) in the master page.
    Label lbl = (Label)Master.FindControl("lblMaster");
    lbl.Text = "From WebForm Page...";
}
```

13. Run the WebForm and Click on Button to see that the text in master page Label has changed.
14. To the class in MainMaster.master.cs add the following property.

Using Label

```
public Label MasterLabel
{
    get { return lblMaster; }
}
```

15. To the Default.aspx add the following

    `<%@ MasterType  VirtualPath="~/MainMaster.master" %>`
16. Replace the existing code in btnDemo_Click with the following

Setting the text in MasterPage

```
//To set Text of Label in master page using the public property MasterLabel
Master.MasterLabel.Text = "From WebForm";
//The above line would work only if <%@MasterType Directive is added to current page
```

URL’s in Master Pages

One potential issue with placing master and content pages in different folders involves broken URLs. If the master page contains relative URLs in hyperlinks, images, or other elements, the link will be invalid for content pages that reside in a different folder. In this tutorial we examine the source of this problem as well as workarounds

ResolveClientUrl returns a path relative to the current page, and **ResolveUrl** returns a path relative to the site root. Both methods are particularly useful when passing in a relative URL prefaced with the tilde (~) to indicate the application root.

Page.ResolveUrl("~/images/demo.gif") renders "/images/demo.gif"

Page.ResolveClientUrl("~/images/demo.gif") renders ".. /images/demo.gif"

Automatic URL Resolution in the <head> Section

While the <link> element's href attribute is relative, it's automatically converted to an appropriate path at runtime. The <head> region is actually a server-side control, which enables it to modify the contents of its inner controls when it is rendered.

ClientIDMode and Importance of UniqueID and ClientID

If a Control is placed in a webform without masterpage:

The id and name attribute rendered to the browser are same as Id of control on server and thus the same Id / Name can be used for programming a the control in JavaScript

If a Control is placed in a webform with masterpage:

The ClientId and Name will not be same as Id of control on server because it might end up in ambiguity if the control with same Id is present is both Master Page and Content Page.

Control.ClientIdMode =

- **AutoID**: This value will make the ClientID generation same as earlier versions of ASP.Net
- **Static**: Setting this value will make the ClientID same as the ID on server. It will not concatenate ID of the parent naming containers.
- **Predictable**: This will be useful to predict the ClientID of child controls in data controls. Setting this property will prevent the “ctlxxx” prefix in the ClientID.
- **Inherit**: This value will make the control to inherit the parent control’s setting. This is the default value for this property.

Predictable Example

For example, if we have child controls inside a ListView control the ClientID will be generated like below by default.

<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1"

ClientIDMode="**Predictable**" **ClientIDRowSuffix**="EmpName">

The above setting will remove the prefix "ctlxx" and will append the value of the column specified in ClientIDRowSuffix. This will generate the ClientID's like,

For each Employee in the dabase following section will be rendered:

```
<div id="ListView1_itemPlaceholderContainer" style="font-family: Verdana, Arial, Helvetica, sans-serif;">
      EmpNo:  <span id="ListView1_EmpNoLabel_Sandeep">2</span>
      EmpName: <span id="ListView1_EmpNameLabel_Sandeep">Sandeep</span>
      Address:  <span id="ListView1_AddressLabel_Sandeep">3rd Cross</span>
      City:  <span id="ListView1_CityLabel_Sandeep">Bangalore</span>
      Country: <span id="ListView1_CountryLabel_Sandeep">US</span>
</div>
```

Example with ClientID and UniqueID:

1. Add a TextBox and HTML Button to webform / content page which has master page.

```
<asp:TextBox runat="server" ID="txtWebForm"/>
<input type="button" value="ClientButton" onclick="ShowAlert()" />
```

2. Add the follwing to the webform (cphHead is Id of ContentPlaceHolder added to Head section of Master Page)

Adding code to MasterPage Head section

```
<asp:Content ID="cntHead" ContentPlaceHolderID="cphHead" runat="server">
  <script language="javascript" type="text/javascript">
  function ShowAlert()
  {
    txt = document.<%=Page.Form.Name%>.<%=txtWebForm.UniqueID%>
    alert(txt.value);
  }
  </script>
</asp:Content>
```