

# Chapter -4 Styles, Themes and Master Pages

## Style Sheets

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language.

### 4.1 Creating Style Sheets

#### How to create Style Sheet in notepad

Notepad is one of the text editors that you can use to create a Cascading Style Sheet (CSS) document. After you create a CSS file from the Notepad, you can link this file to your Web pages so that the content of your Web pages can be formatted by your style sheet.

**Step 1:** Click on Start Menu → Accessories → Notepad

**Step 2:** Type your code.

**Step 3:** Save the Notepad file, by clicking "File" and select "Save." Option. A "Save As" dialog box is displayed. Assign appropriate name and save file with .css extension.

#### CSS Rules

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector:** A selector is an HTML tag at which style will be applied. This could be any tag like <h1> or <table> etc.
- **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color or border etc.
- **Value:** Values are assigned to properties. For example color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule **Syntax** as follows:

**Syntax:**

```
selector { property: value }
```

You can define a table border as follows:

**Example:**

```
table{ border :1px solid #C00; }
```

Here table is a selector and border is a property and given value 1px *solid* #C00 is the value of that property and it is called HEX code.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

## CSS Selectors

### The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

**Example**

```
p {
  text-align: center;
  color: red;
}
```

**The Universal Selectors:** Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type:

**Example:**

```
* {
  color: #000000;
}
```

This rule renders the content of every element in our document in black.

**The Class Selectors:** You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

**Example:**

```
.black {  
    color: #000000;  
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular.

**Example:**

```
h1.black {  
    color: #000000;  
}
```

This rule renders the content in black for only <h1> elements with class attribute set to black. You can apply more than one class selectors to given element. Consider the following example:

**Example:**

```
<p class="center bold">  
This para will be styled by the classes center and bold.  
</p>
```

### 4.1.1 Applying Style Sheets

A .css file is a Cascading Style Sheet file that contains the style, behavior about a html page and its elements. These styles are written in the .css file as a block of code and they are referred to as css class

CSS can be added to HTML in the following ways:

- Inline - using the style attribute in HTML elements
- Internal - using the <style> element in the <head> section
- External - using an external CSS file

## 1. Inline Styles

An inline style can be used if a unique style is to be applied to one single occurrence of an element.

To use inline styles, use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example below shows how to change the text color and the left margin of a paragraph:

```
<p style="color: blue; margin-left: 20px ;"> This is a paragraph. </p>
```

HTML Style Example - Font, Color and Size

The font-family, color, and font-size properties define the font, color, and size of the text in an element:

### Example

```
<!DOCTYPE html>
<html>
<body>
<h1 style="font-family:TimesNewRoman;">A heading</h1>
<p style="font-family:arial;color:blue;font-size:15px;">A paragraph.</p>
</body>
</html>
```

## 2. Internal Style Sheet

An internal style sheet can be used if one single document has a unique style. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

### Example

```
<head>
<style>
body {background-color:blue;}
p {color:blue;}
</style>
</head>
```

## 3. External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the <head> section:

**Example:**

```
<head>
<link rel="style sheet" type="text/css" href="mystyle.css">
</head>
```

## HTML Style Tags

Tag	Description
<style>	Defines style information for a document
<link>	Defines the relationship between a document and an external resource

## 4.2 Theme

ASP.NET Themes and ASP.NET skins are another method of adding style and consistency to ASP.NET pages without managing each page and control separately. Themes and skins are an alternative to using Master Pages.

A theme is a collection of property settings that allow you to define the look of pages and controls, and then apply the look consistently across pages in a Web application, across an entire Web application, or across all Web applications on a server.

Themes are made up of a set of elements: skins, cascading style sheets (CSS), images, and other resources. At a minimum, a theme will contain skins. Themes are defined in special directories in your Web site or on your Web server.

ASP.NET themes are a unified look which can be applied to every page in a web site. ASP.NET skins are also like a theme, but it can be applied to Controls like Buttons Controls, DropDownList Controls.

### What is .skin file?

A skin is a definition of styles applied to the server controls in your ASP.NET page. Skins can work in conjunction with CSS files or images. The skin file can have any name, but it must have a .skinfile extension.

.skin file is a new type of file introduced in ASP.NET 2.0, this can contain style information of any asp.net server control. These information are written in this file as if they are written asp.net page but ID property is not specified. for example

**Example:**

```
<asp:Label runat="server" Font-Names="Courier" />
<asp:Label SkinID="LabelMessage" runat="server" ForeColor="Green"
BackColor="Yellow" />
<asp:Label SkinID="LabelError" runat="server" ForeColor="Red"
BackColor="Yellow" />
```

In above code id of the control. Also the SkinId is optional to specify. If you have used the theme with above skin in your project, the 1st line defines the style of all Label controls that will be used in the aspx page. 2nd line defines the style of only those label controls whose SkinID is specified as **LabelMessage**. In this way you have liberty of placing number of Label controls in your .skin files with different SkinID for different types of use. (Like you can define a skin for Label control to show success message and another skin to show error message on the page.

If you want the exact behavior of the Label control as you have defined for SkinID "LabelMessage", you should be using the SkinID value of your Label control in the aspx page as "LabelMessage".

The behavior of the Label control in your aspx page will differ based on whether you have specified the SkinID of the Label control or not or specified the CssClass or not.

There is hardly any limitation of number of .css and .skin files you can create in a Theme but it is suggested to limit the number for easier maintainability and tracking.

**Example of Skin**

The Summer.skin file.

```
<asp:Label Runat="server"
    ForeColor="#004000"
    Font-Names="Verdana"
    Font-Size="X-Small" />
```

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

<asp:Button Runat="server"
    ForeColor="#004000"
    Font-Names="Verdana"
    Font-Size="X-Small"
    BorderStyle="Solid"
    BorderWidth="1px"
    BorderColor="#004000"
    Font-Bold="True"
    BackColor="#FFE0C0" />
```

Control definitions must contain the Runat="server" attribute. No ID attribute is specified in the skinned version of the control. Using the theme in an ASP.NET page.

#### **Example of using theme in an ASP.NET Page**

```
<%@ Page Language="VB" Theme="Summer" %>

<script runat="server">
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        Label1.Text = "Hello " & Textbox1.Text
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
```

```
<title>INETA</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:Textbox ID="TextBox1" Runat="server">
    </asp:Textbox>
    <asp:Button ID="Button1"
      Runat="server"
      Text="Submit Your Name"
      OnClick="Button1_Click" />
    <asp:Label ID="Label1" Runat="server" />
  </form>
</body>
</html>
```

## 4.2.1 How Themes Work

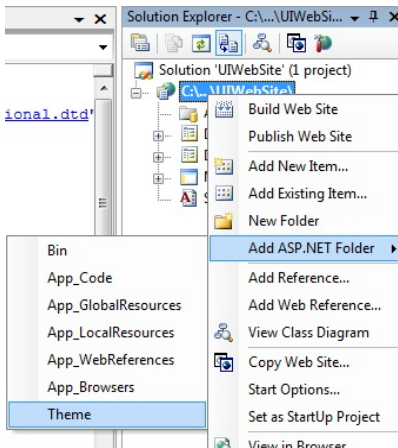
### 4.2.1.1 Creating themes

In order to use ASP.NET Themes, you should first create ASP.NET Theme.

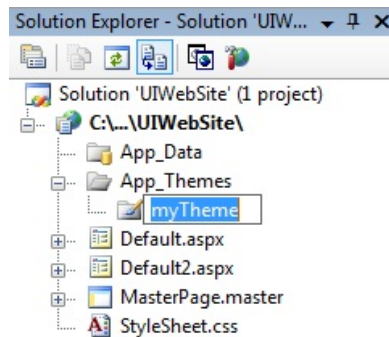
ASP.NET by default searches for available themes in a folder called App\_Themes . The following steps show you how to create this folder and start a new theme, and design skins for two ASP.NET Controls.

1. Right click the Project name in the Solution Explorer and select  
Add ASP.NET Folder - -> Theme from the Context Menu.



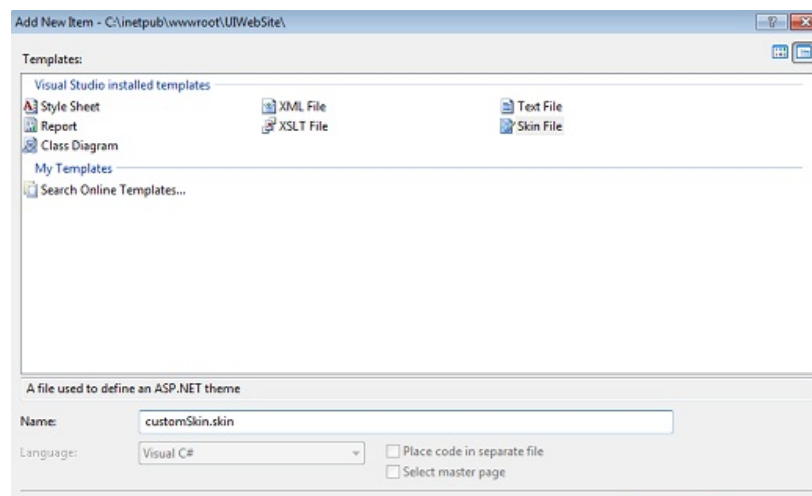


Adding App\_Theme Folder, Rename Theme1 to myTheme.



Renaming Custom ASP.NET Theme

2. Right Click the myTheme folder and add a Skin File named customSkin.skin. The customSkin.skin file will open with the default skin template.



Adding customSkin.skin File

At the bottom of the customSkin.skin File, after the --%> markup, enter the following skin description code.

**Code:**

```
<asp:DropDownList runat="server" CssClass="myCustomStyle" >
</asp:DropDownList>

<asp:Button runat="server" BorderStyle="Dotted" CssClass="myCustomStyle"
/>
```

Right Click on the myTheme folder and Add a new Style Sheet named myCustom.css to the Project.

Add the following Style Class into the myCustom.css.

**Style Class**

```
.myCustomStyle
{
    background-color:#ccffcc;
    font-style: italic;
```

```
font-family: Arial, Tahoma, 'Trebuchet MS';  
font-size: xx-large;  
}
```

Now you have successfully created a skin for an ASP.NET DropDownList Control and a Button Control. The content of the customSkin.skin declares that all controls of these types should use the CssClass called myCustom.css. The myCustomStyle class is in the myCustom.css.

The Button control has been programmed to use a dotted border style in the customSkin.skin. This property isn't part of the style sheet and therefore applies only to buttons and not to the DropDownList control.

#### 4.2.1.2 Applying theme to a website

The best and the fastest way to tell every page and every control in a web site to use a given theme is to code it in the web.config file. The following steps show you how to make a theme available site wide.

- Open the web.config file from the Solution Explorer.

**Note:**

If you do not have a web.config file, run the application for the first time, and it will ask whether to create one or not.

- Go to the element starting with <pages within the <system.web> section.

**Note:**

If the web.config file does not have <pages just type in this tag, and Visual Studio will suggest and help you auto fill it.

- Add the following code snippet after the <pages for the theme attribute.

**Code:**

```
<pages theme="myTheme">  
    My Page  
</pages>
```

At runtime, every page in the web site now knows to use the myTheme ASP.NET theme. This ASP.NET theme makes the DropDownList and the Button Controls within the pages to use the myTheme ASP.NET theme styles.

- You also can use the `StyleSheetTheme` attribute instead of the `theme` attribute. The `StyleSheetTheme` attribute allows local setting to override the global theme.

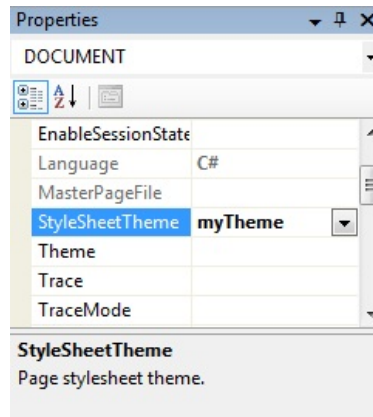
**Code:**

```
<pages StyleSheetTheme="myTheme">  
    My Page  
</pages>
```

#### 4.2.1.3 Applying an ASP.NET theme to a page

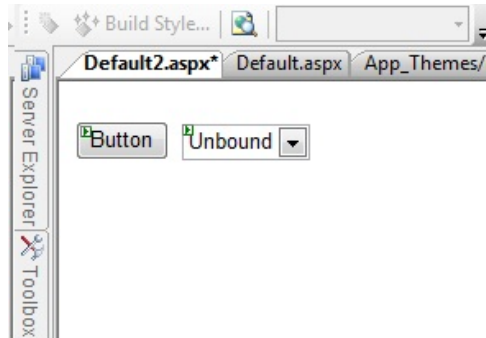
You also have the capability of setting a theme for an individual page alone by configuring its `theme` or `StyleSheetTheme` settings. This allows different themes on different pages. The following steps show you how to apply a theme to an individual page.

1. Remove any themes assigned in the `web.config` file. (Check Assigning a theme to the whole Web Site). This is because if you leave the theme declared in the `web.config` file, it effects for whole site, thus it overrides any theme applied to an individual page locally.
2. Go to the Design View of the ASP.NET page.
3. Select the page by clicking on the blank area and go to the Properties Window of the Document Object (You can also press F4).
4. Set the `StyleSheetTheme` property to the name of you preferred theme (`myTheme`).

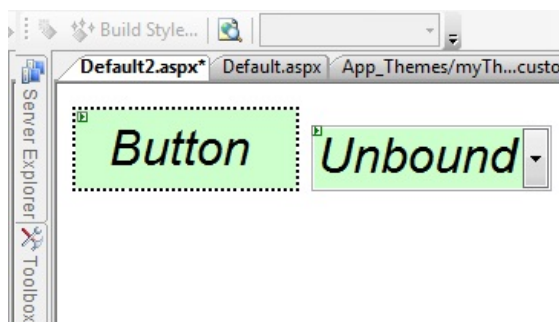


### StyleSheet Theme Property

Drag and drop a DropDownList and a Button Control on the page and you will see how the IDE looks up themyTheme ASP.NET theme and applies the style information in the Design View.



### Controls without theme



**Custom Controls with theme**

### 4.2.2 Creating Multiple Skins for the same control

To create multiple skins for single element, you use the SkinID attribute to differentiate among the definitions.

#### Example of Multiple Skins

File: Summer.skin file

```
<asp:Label Runat="server" ForeColor="#004000" Font-Names="Verdana"
    Font-Size="X-Small" />

<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="Verdana"
    Font-Size="X-Small" BorderStyle="Dotted" BorderWidth="5px"
    BorderColor="#000000" Font-Bold="False" SkinID="TextboxDotted" />

<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:Button Runat="server" ForeColor="#004000" Font-Names="Verdana"
    Font-Size="X-Small" BorderStyle="Solid" BorderWidth="1px"
    BorderColor="#004000" Font-Bold="True" BackColor="#FFE0C0" />
```

A simple .aspx page that uses the Summer.skin file with multiple text-box style definitions

**Example of applying multiple skins to same control**

```
<%@ Page Language="VB" Theme="Summer" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Different SkinIDs</title>
</head>
<body>
  <form id="form1" runat="server">
    <p>
      <asp:Textbox ID="TextBox1" Runat="server">Textbox1</asp:Textbox>
    </p><p>
      <asp:Textbox ID="TextBox2" Runat="server"
        SkinId="TextboxDotted">Textbox2</asp:Textbox>
    </p><p>
      <asp:Textbox ID="TextBox3" Runat="server"
        SkinId="TextboxDashed">Textbox3</asp:Textbox>
    </p>
  </form>
</body>
</html>
```

### 4.3 Master Page

Master pages provide templates for other pages on your web site.

ASP.NET master pages enable you to create a page layout (a master page) that you can use with selected or all pages (content pages) in your Web site. Master pages can greatly simplify the task of creating a consistent look for your site.

A **master page** provides a template for other pages, with shared layout and functionality. The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page. Extension of Master-Page is '.master'.

We cannot run Master-Page directly.

The **content pages** contain the content you want to display.

When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.

### 4.3.1 Characteristics of Master Page

- Master page also have code behind file
- We cannot run masterpage directly
- It allows you to centralize the common functionality of your pages so that you can make updates in just one place.
- A Masterpage has 2 parts.
  1. Content that appears on each page that inherits the master page.
  2. Regions that can be customized by the pages inheriting master page.
- A Master page need to specify both parts common to all pages and the parts that are customizable
- Items you add to the master page appear on all pages that inherit it.

### ContentPlaceholder

A control that should be added on the MasterPage which will reserve the area for the content pages to render their contents. Content place holders are used to allocate places on the MasterPage that will be changed on the content pages.

### Content Control

A control which will be added on content pages to tell these pages that the contents inside this control will be rendered where the MasterPage's ContentPlaceholder is located.

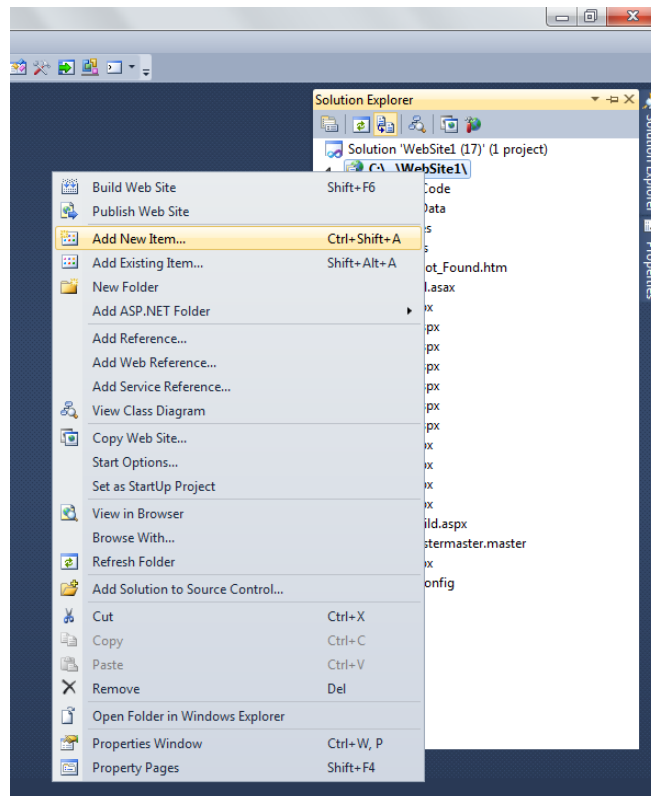
### 4.3.2 Basics of Master Pages



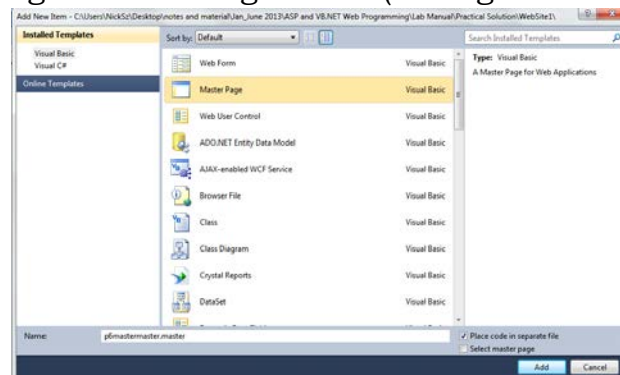
## Creating Master Page

### Steps :

1. Solution Explorer → Root Directory → Right Click → Add New Item or Press Ctrl+Shift+A



2. Select Master Page From dialogue box. (Change the name if you wish)



### 3. Design/Code the master page as per web page's need.

Here we have made some designing in the master page named as pmastermaster.master

#### Design View :



**Note :** The empty row between Menu & Footer is ContentPlaceHolder, where child pages are included automatically. Master page can't be viewed in a browser itself, so empty child page is included & displayed here.

#### Design Code : (Master Page)

```
<%@ Master Language="VB" CodeFile="pmastermaster.master.vb"
Inherits="pmastermaster" %>
<!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></title>
  <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
```

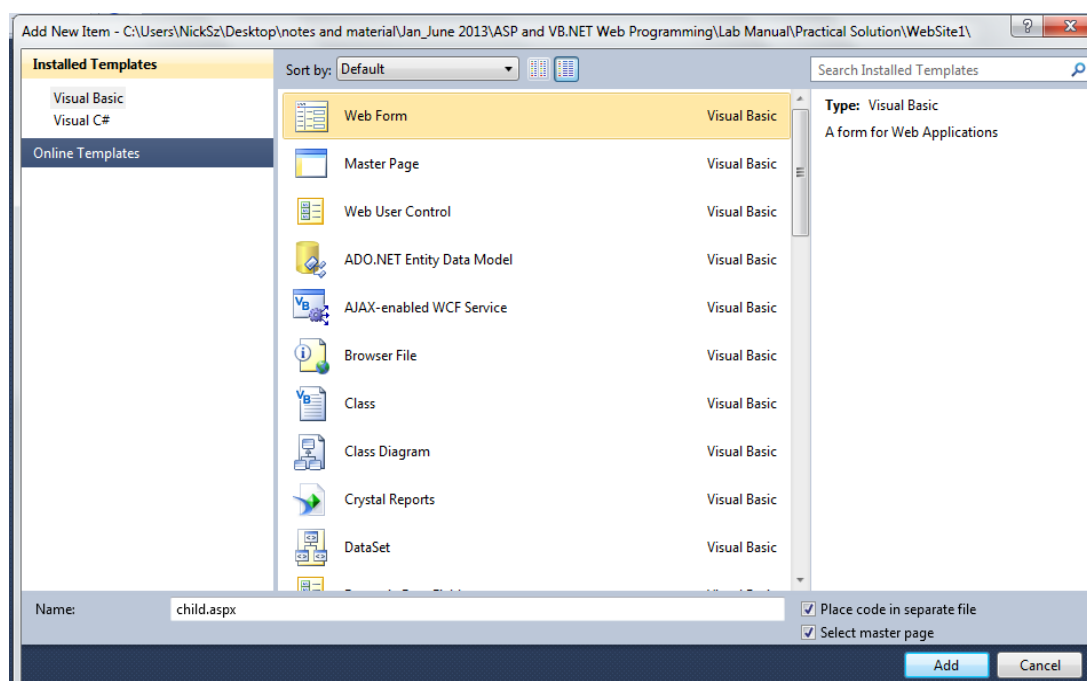
```
<body>
  <form id="form1" runat="server">
    <table border="1" rules="all" style="width: 500px;margin left:25%;margin-
top:100px;">
      <tr>
        <td align="center">

        </td>
        <td style="font-size: 30px; font-weight: bold; text-align: center">
          Android World
        </td>
      </tr>
      <tr>
        <td colspan="2" style="font-size: 15px; font-weight: bold; text-align:
center">
          Menu.....
        </td>
      </tr>
      <tr>
        <td colspan="2" align="center">
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
<%--Child Page will Reside here--%>
          </asp:ContentPlaceHolder>
        </td>
      </tr>
      <tr>
        <td colspan="2" style="font-size: 15px; font-weight: bold; text-align:
center">
          Footer.....
        </td>
      </tr>
    </table>
  </form>
```

```
</body>
<html>
```

### 4.3.2 How Master Page and Content Pages are connected

- Add new web form using any of the way.
- Choose Web Form from dialogue box & check select master page in it.



#### Example of Child Page created under Master page

##### Design Code : (Child Page)

```
<%@ Page Title="" Language="VB"
MasterPageFile="~/p6mastermaster.master" AutoEventWireup="false"
CodeFile="p6_child.aspx.vb" Inherits="p6_child" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="Server">
```


```

<label style="font-size: 20px; font-weight: bold; text-align: center">
    Child Page Contents Here.....<br />
    Child Page Contents Here.....<br />
    Child Page Contents Here.....<br />
</label>
</asp:Content>

```

### Program Output :



	<b>Android World</b>
Menu.....	
Child Page Contents Here.....	
Child Page Contents Here.....	
Child Page Contents Here.....	
Footer.....	

### 4.3.3 Nesting Master Page

#### Note :

- Nested master page means master page with in a master page.
- There will be two master pages.
- One will act as parent master page & other as child master page
- Child master page is included in the ContentPlaceHolder of parent master page.
- So child master page will automatically inherit the designing of parent master page.
- To use nested mater page one simple rule is applied & is as followed :
- Place the code in ContentPlaceHolder of parent using its ID, from where we want to inherit design/code.

- Here we will create two master pages & a child page for child Master page.
- Steps for creating master page is same as illustrated in Practical VI.
- The remaining are illustrated below.

### Steps :

- Add Master page

Note : Here we have named it as Pmaster.master

#### Design Code : (Master Page – Pmaster.master)

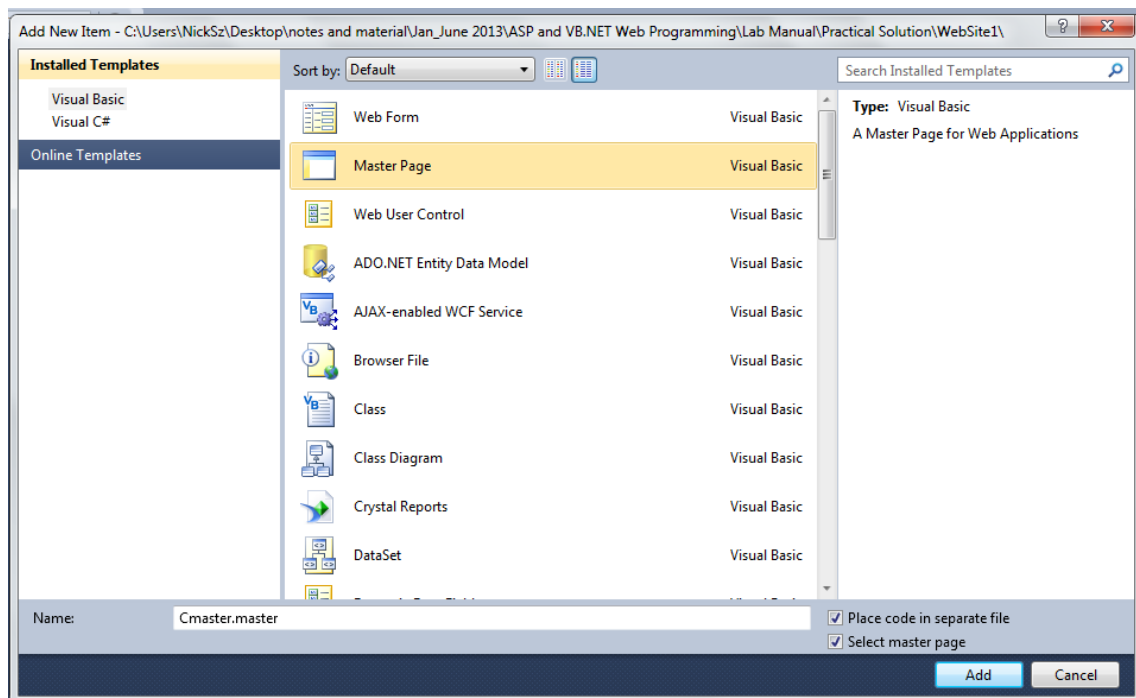
```
<%@ Master Language="VB" CodeFile="Pmaster.master.vb" Inherits="Pmaster"
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
<table>
    <tr>
        <td>
            Header.....
        </td>
    </tr>
    <tr>
        <td>
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
        </td>
    </tr>
    <tr>
        <td>
```

```

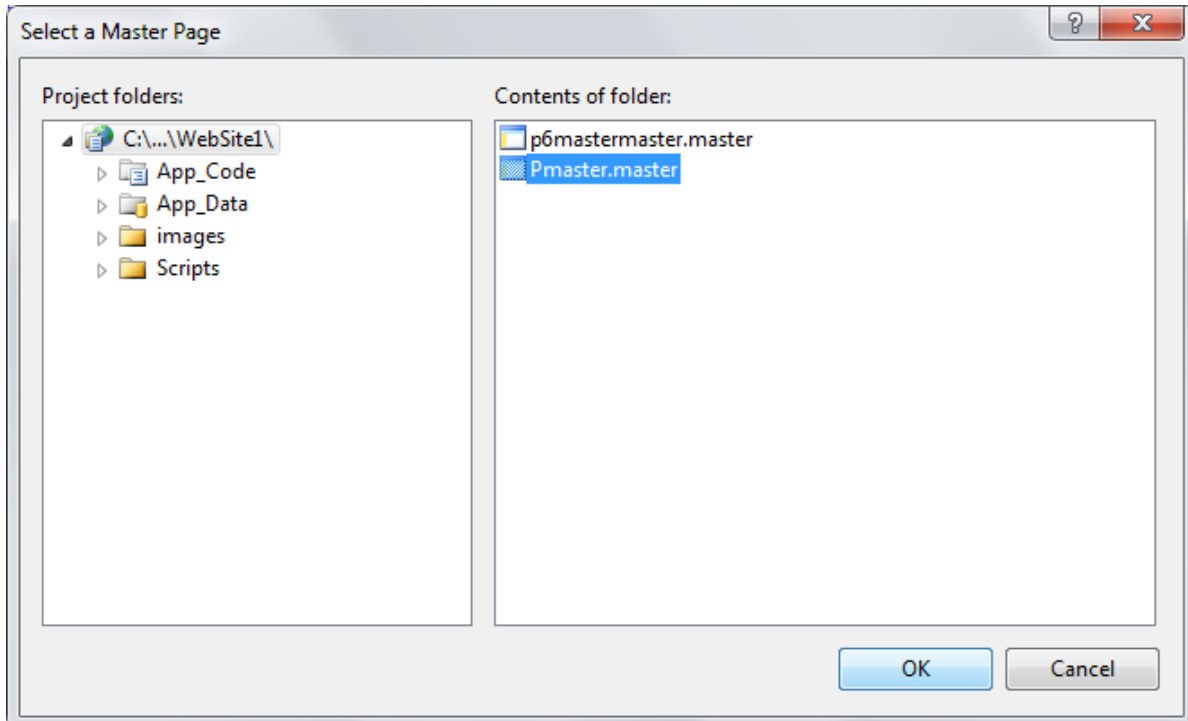
        Footer....
    </td>
</tr>
</table>
</form>
</body>

```

- Add another master page the same way, but in the dialogue box, check select master page.



- Select parent master page (Pmaster.master in this case) & click OK.



- Code the further designing/coding the way you want in child master page.

**Design Code : (Child Master Page – Cmaster.master)**



```
<%@ Master Language="VB" MasterPageFile="~/Pmaster.master"
AutoEventWireup="false"
    CodeFile="Cmaster.master.vb" Inherits="Cmaster" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="Server">
<table>
    <tr>
        <td>
            child master header...
        </td>
    </tr>
    <tr>
        <td>
<asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">
<%-- Child Pages will be included here..... --%>
</asp:ContentPlaceHolder>
        </td>
    </tr>
    <tr>
        <td>
            Child Master footer
        </td>
    </tr>
</table>
</asp:Content>
```

- Add child page,select Cmaster.master as master page.

**Design Code : (Child Page – child\_page.aspx)**

```
<%@ Page Title="" Language="VB" MasterPageFile="~/Cmaster.master"
AutoEventWireup="false" CodeFile="child_page.aspx.vb" Inherits="child_page"
%>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder2"
```

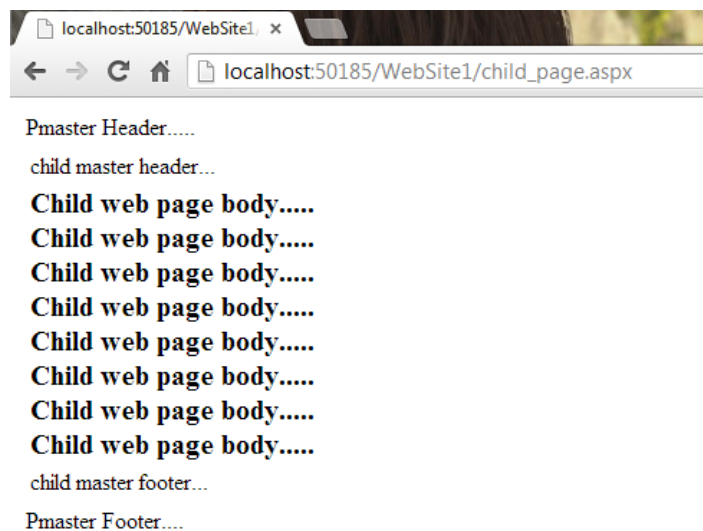
```

Runat="Server">
<label style="font-size: 20px; font-weight: bold; text-align: center">
    Child web page body.....<br />
    Child web page body.....<br />
    Child web page body.....<br />

    Child web page body.....<br />
    Child web page body.....<br />
    Child web page body.....<br />
    Child web page body.....<br />
    Child web page body.....
</label>
</asp:Content>

```

### Program Output :



**Exercise**

1. What is CSS ? Explain its types along with proper example
2. Explain what is theme ? Write down steps to create theme.
3. What is Skin ? Explain it.
4. What is Master Page ? Explain its characteristics.
5. Explain Nested Page with appropriate example.

\*\*\*\*\***END OF CHAPTER**\*\*\*\*\*