UNIT-IV Master Pages and Themes

*Need for Master Pages

Websites always use some elements repeatedly, like ***menu, logos, headers, footers, and sidebars***.

Some developers simply copy and paste the code for the common elements to every page. This works, but the process is very cumbersome **(Difficult and time consuming**).

Every change to any of the elements needs to be done to each page manually. Formerly, this was the only way to duplicate the change.

Master pages are created in a website to enhance modularity and user-friendliness.

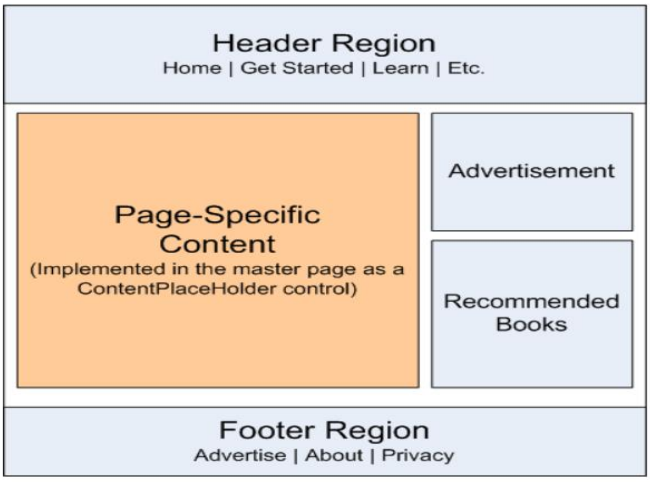
They help standardize the designing elements like header, footer, navigation menus, and other parts, which are common to the entire web site and present on every single page of the web application.

Definition- *Master Pages provides consistent and same page layout to multiple webpages in website.*

Master page having extension **.master**

Master page uses **@Master** Directive in source code which provides page specific information

*Master page can have zero or more **ContentPlaceHolder** Controls.*



General Layout for a master page

Basics of Master Pages-

- **Masterpage**: Gives us a way ***to create common set of UI elements*** that are required ***on multiple pages of our website***.
- **ContentPage**: The ***ASP.NET web page that will use master page*** to have the common UI elements displayed on rendering itself.
- **ContentPlaceHolder**: A control that should be added on the Master Page ***which will reserve the area for the content pages to render their contents.***
- **ContentControl**: A control which will be added on content pages to tell these pages that ***the contents inside this control will be rendered where the Master Page's ContentPlaceHolder is located.***

Advantages of Master Page-

1) To get standard uniformity among all webpages in website instead given it to each page separately.

2) To achieve easiness of global changes i.e. modification or editing can be done only in master page which will automatically replicates in all content pages in website.

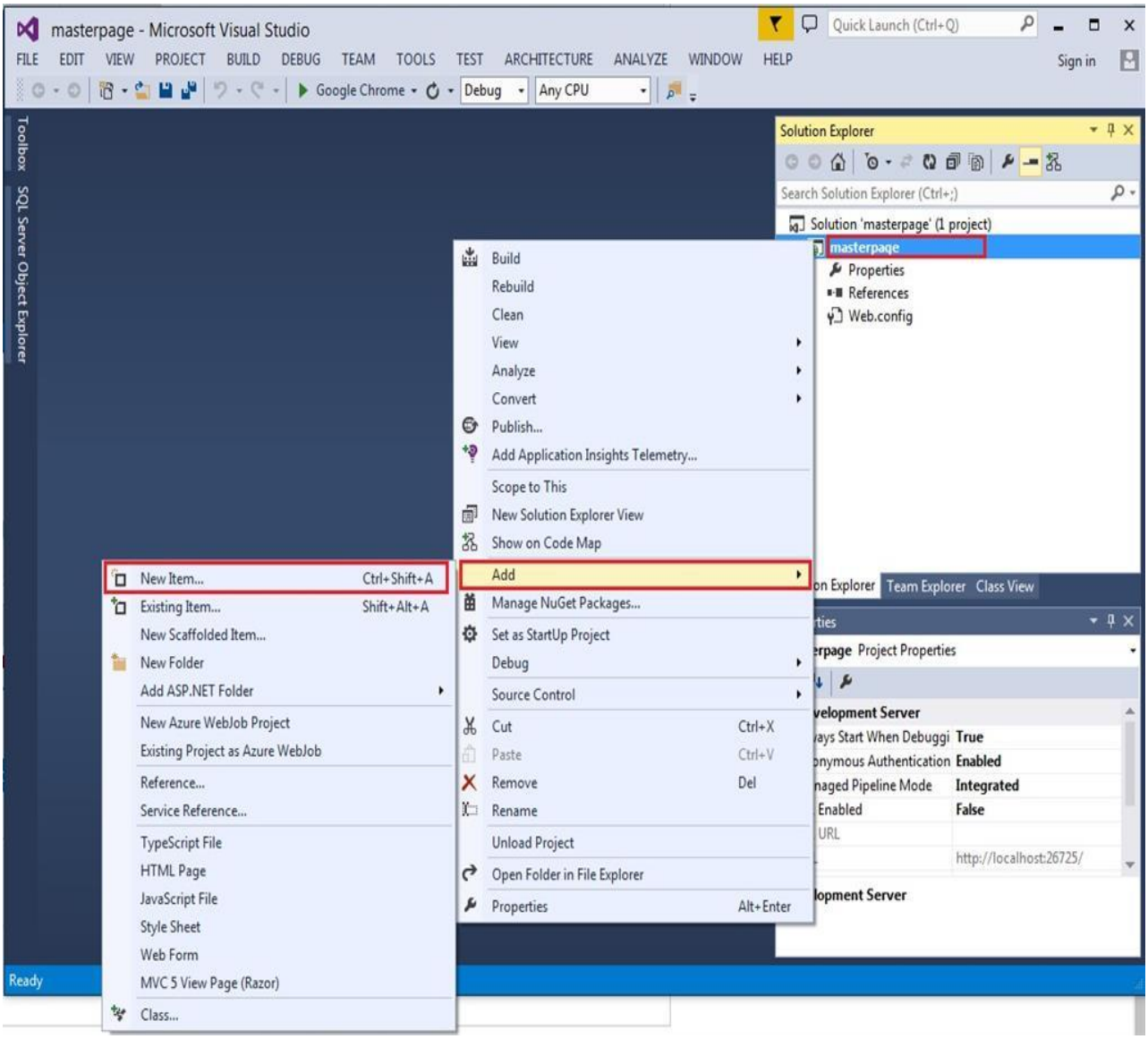
3) Time Saving for UI development.

Creating Master And Content Pages-

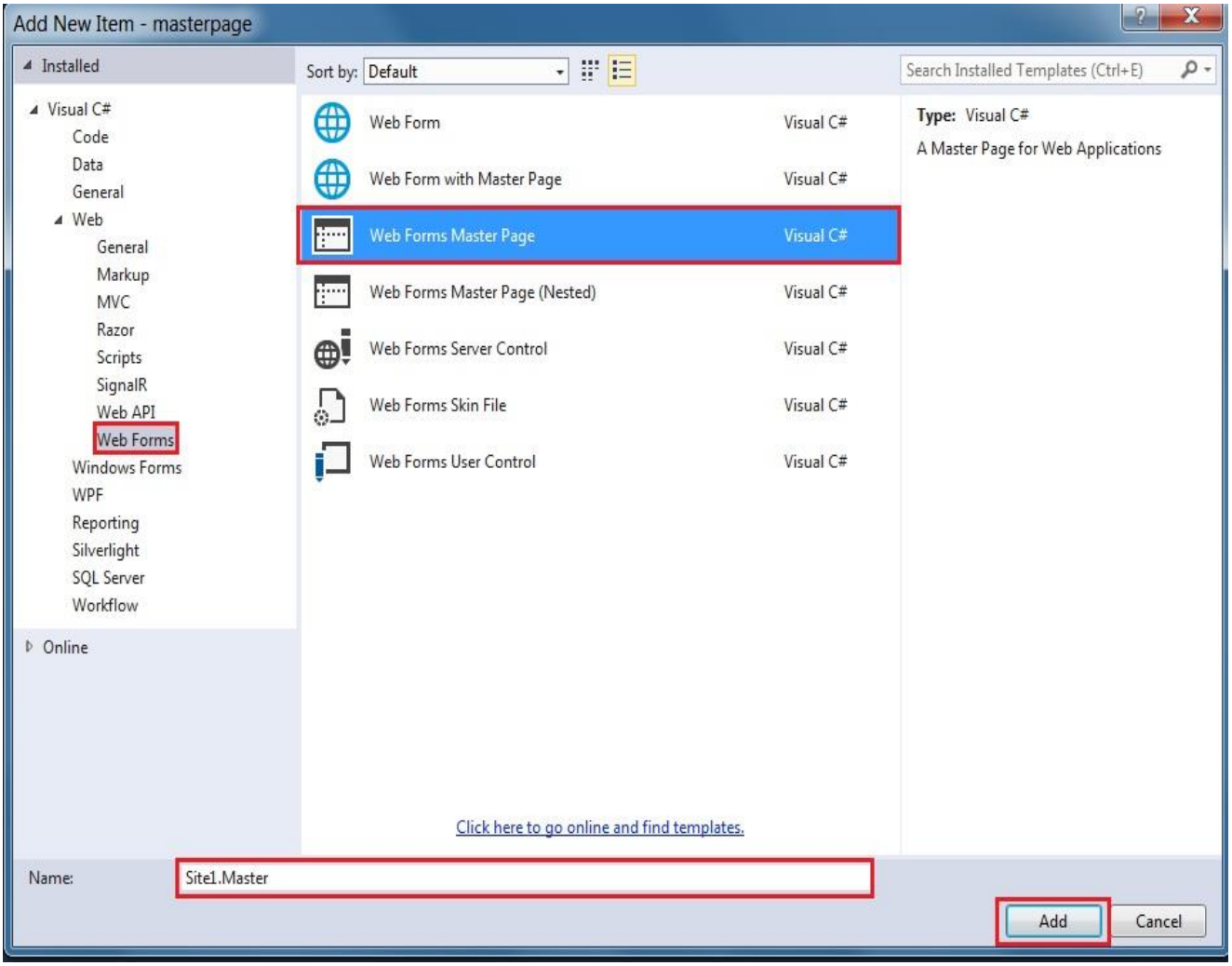
Step 1: Add new file in to our project.

Add the master page into our project.

Right click Project->Add->New item (shown in the picture),

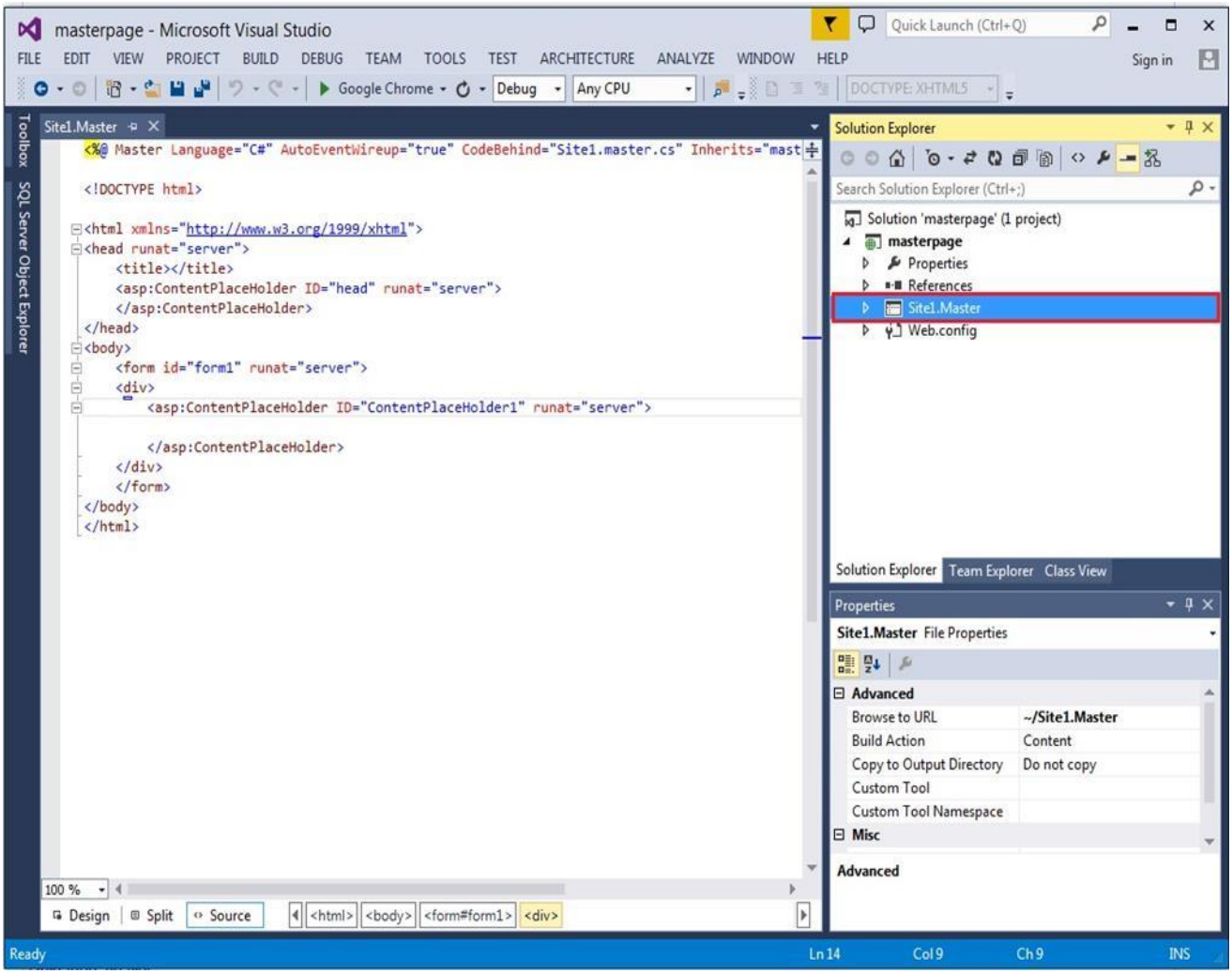


After clicking on new item, Window will open, select Web Form->Web Forms Master Page (shown in the picture),



After clicking the add button, master page 'site1.master' adds to our project.

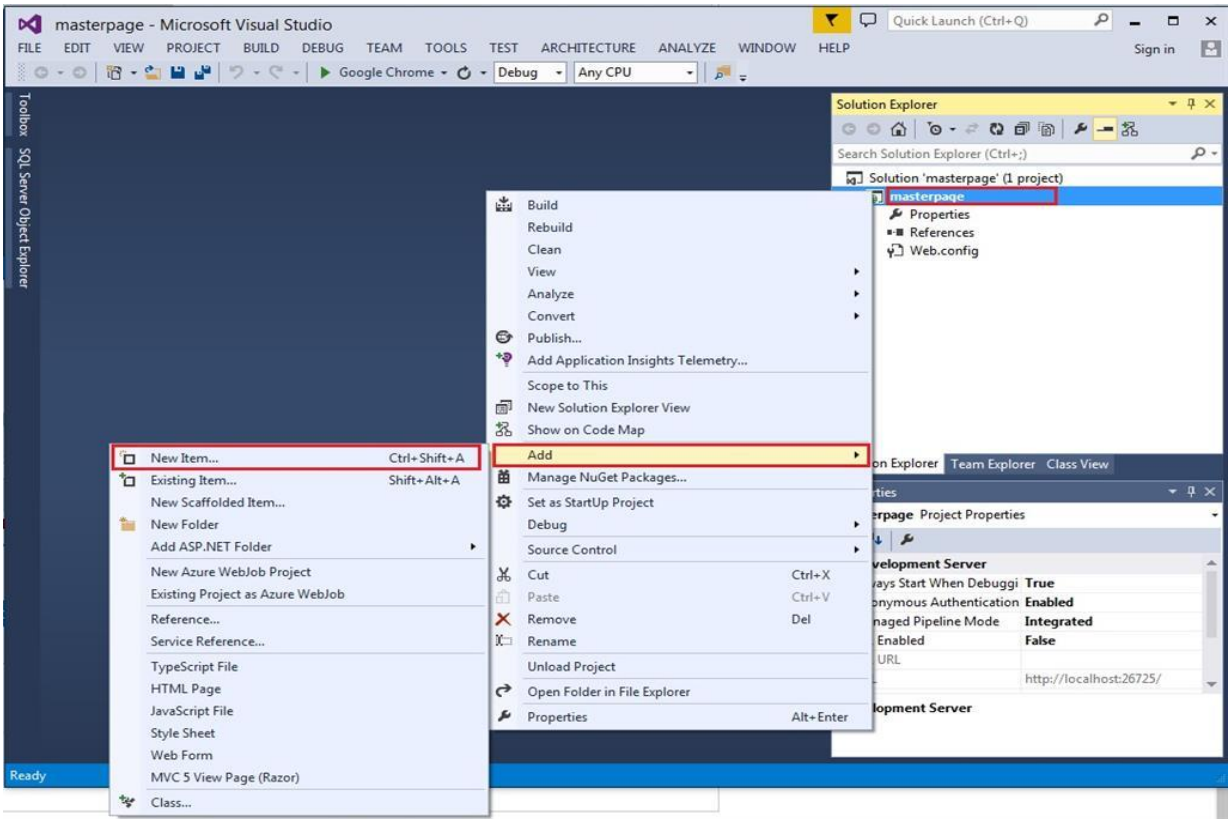
Click on site1.master into Solution Explorer (shown in the picture),



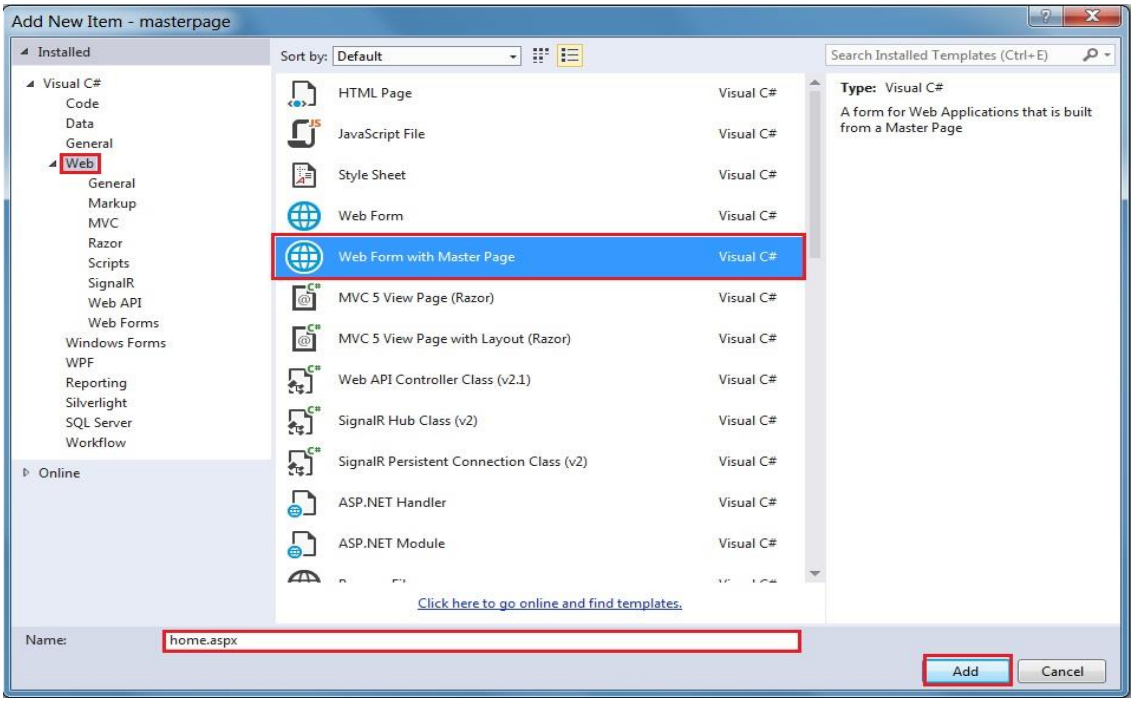
Step 2: Design the master page, using HTML.

Step 3: **A**dd web form in to our project.

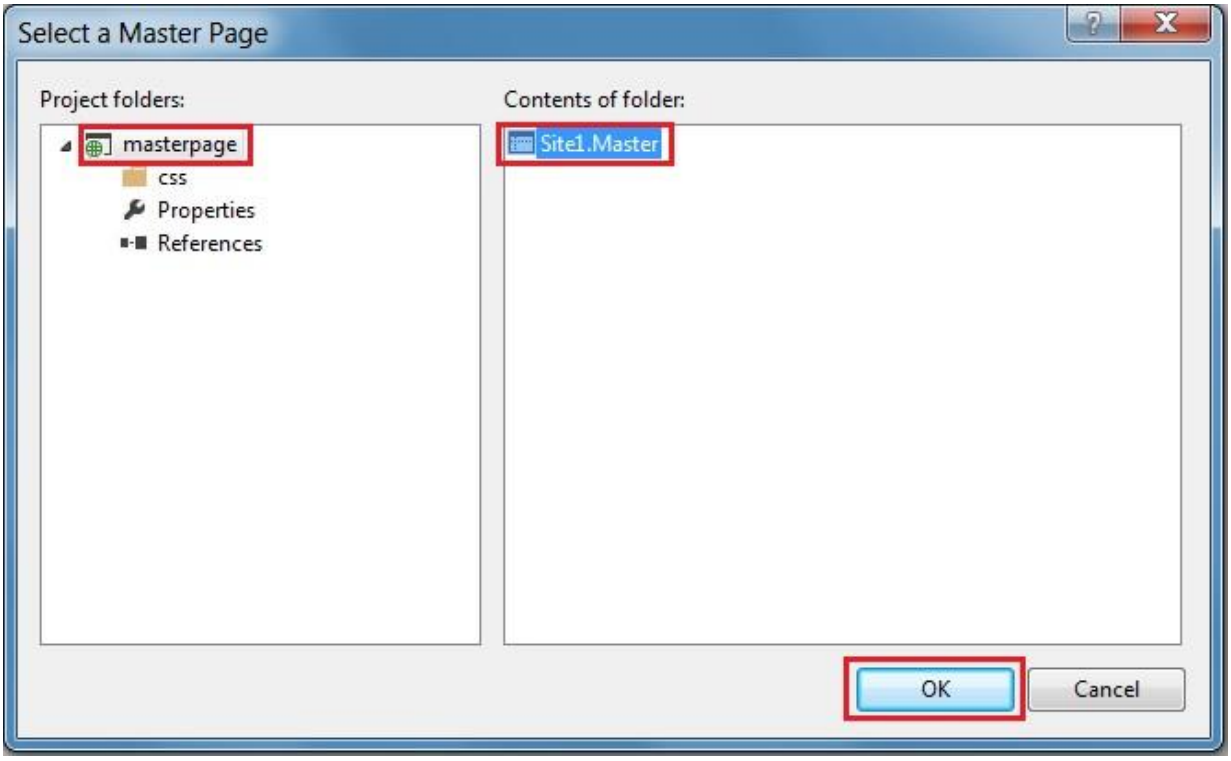
Right click on the project->Add->New item (shown in the picture),



Select Web form with the master page.



After clicking on that, add the button Window, open the selected masterpage->site1.master and click OK.



Now, design our homepage.

Here, we write home page only,

Home.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEvent
Wireup="true" CodeBehind="home.aspx.cs" Inherits="masterpage.home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" run
at="server">
   <h1>Home page</h1>
</asp:Content>
```

Finally, our Master page is created; build and run the project.

Programmatically assign Master Pages to Web Page-

You have an application where you need to change the assignment of the master pages used at runtime, can use following way

In the **Page_ PreInit** event handler of code-behind for the **content pages**, set the **Page.MasterPageFile** property to the name of the desired **.master** file.

```
protected void Page_PreInit(object sender, EventArgs e)
{
    this.MasterPageFile = "MyMainMasterPage.master";
}
```

Note—You have to add Content element in source code of web page.

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">

</Content>
```

Nested Master Pages-

When one master page references another as its master, it is said to be a nested master page.
A number of nested masters can be componentized into a single master.

There is no architectural limitation to the number of child masters that can be created. This nested model is pretty cool and powerful, and allows arbitrary levels of nesting.
The depth of nesting also does not impact on performance significantly.

The advantage of this kind of structuring is that a number of child masters can be created to be subordinated to the overall look and feel of the site defined by the Parent master, while the child master give the child pages some uniqueness. While a parent master defines the overall layout of the pages **header, body and footer**, the **child master expands the body for a group of pages.**

Just like the parent master page child masters have the extension **.master** .

It contains all the controls that are mapped to content place holders on the parent master page.

The layout is similar to that of a content page in this respect. However, ***child masters also have content place holders of their own to display content of its child pages.***

Event Ordering in Master Pages-

The following is the sequence in which events occur when a master page is merged with a content page:

1. Content page PreInit event.
2. Master page controls Init event.
3. Content controls Init event.
4. Master page Init event.
5. Content page Init event.
6. Content page Load event.
7. Master page Load event.
8. Master page controls Load event.
9. Content page controls Load event.
10. Content page PreRender event.
11. Master page PreRender event.
12. Master page controls PreRender event.
13. Content page controls PreRender event.
14. Master page controls Unload event.
15. Content page controls Unload event.
16. Master page Unload event.
17. Content page Unload event.

Basics of Themes and Skins-

A Theme decides the look and feel of the website.

Definition - ***Theme is a collection of files that define the looks of a page.***

It can include ***skin files, CSS files & images***.

We define themes in a special **App_Themes** folder. Inside this folder is one or more subfolders named Theme1, Theme2 etc. that define the actual themes.

The theme property is applied late in the page's life cycle, effectively overriding any customization you may have for individual controls on your page.

Skins

A skin file has the file name extension .skin and contains style property settings for individual controls used on web page.

So we can define property settings for controls used on web page such as Button, Label, TextBox, or Calendar etc. controls.

Control skin settings are like the control markup itself, but contain only the properties you want to set as part of the theme.

For example, the following is a control skin for a Button control:

```
<asp:button runat="server" BackColor="lightblue" ForeColor="black" />
```

You create .skin files in the Theme folder. A .skin file can contain one or more control skins for one or more control types. You can define skins in a separate file for each control or define all the skins for a theme in a single file.

There are two types of control skins, default skins and named skins:

- A **default skin** automatically applies to all controls of the same type when a theme is applied to a page.

  A control skin is a default skin if it does not have a SkinID attribute.

  For example, if you create a default skin for a Calendar control, the control skin applies to all Calendar controls on pages that use the theme. (Default skins are matched exactly by control type, so that a Button control skin applies to all Button controls, but not to LinkButton controls or to controls that derive from the Button object.)

Example- MySkinFile.skin

- ***A named skin is a control skin with a SkinID property set.***

  Named skins do not automatically apply to controls by type. Instead, you explicitly apply a named skin to a control by setting the control's SkinID property.

  Creating named skins allows you to set different skins for different instances of the same control in an application.

Example- MySkinFile.skin

x.cs | MySkinFile.skin* | one.aspx

```
<asp:Label runat="server" SkinID="one" BackColor="Red" ForeColor="Green" Font-Name="Monotype Corsiva" />

<asp:TextBox runat="server" BackColor="Black" ForeColor="White" Font-Name="Times New Roman" Width="300px"
Height="70px"/>

<asp:Button runat="server" BackColor="Blue" ForeColor="White" Font-Name="Algerian" Font-Size="18"/>

<asp:Label runat="server" SkinID="two" BackColor="Pink" ForeColor="Red" Font-Name="Calibri"  />
```

Here in above named skin file two seprate style properties are defined for Label Control which can be differenceiate using SkinID Property.

There are 3 different options to apply themes to our website:

1. ***Setting the theme at the page level***:

   The Theme attribute is added to the page directive of the page to apply theme to that page .

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="
Default.aspx.cs" Inherits="Default" Theme="Theme1"%>
```

2. ***Setting the theme at the Application (website) level:*** *to set the theme for the entire website* you can set the theme in the **web.config** of the website.

   Open the web.config file and locate the **<pages>** element and add the theme attribute to it:

```
1. <pages theme="Theme1">
2. ....
3. ....
4. </pages>
```

3. ***Setting the theme programmatically at runtime***:

   here the theme is set at runtime through coding. It should be applied earlier in the page's life cycle i.e. **Page_PreInit event** should be handled for setting the theme.

```
protected void Page_PreInit(object sender, EventArgs e)
   {
                    Theme = "MySkinFile";
   }
```

Uses of Themes

1. Since themes can contain CSS files, images and skins, you can change colors, fonts, positioning and images simply by applying the desired themes.

2. You can have as many themes as you want and you can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also you can switch between themes programmatically.

3. Setting the themes programmatically, you are offering your users a quick and easy way to change the page to their likings.

4. Themes allow you to improve the usability of your site by giving users with vision problems the option to select a high contrast theme with a large font size.