Unit: 4 – Master Pages, Theme Caching, Application Pages & Data

What is Master Page?

A master page is an ASP.NET file with the extension. master (for example, MySite.master) with a predefined layout that can include static text, HTML elements, and server controls. The master page is identified by a special @ Master directive that replaces the @ Page directive that is used for ordinary .aspx pages. The directive looks like

<%@ Master Language="C#" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

Requirement of a Master Page in an Asp.NET application

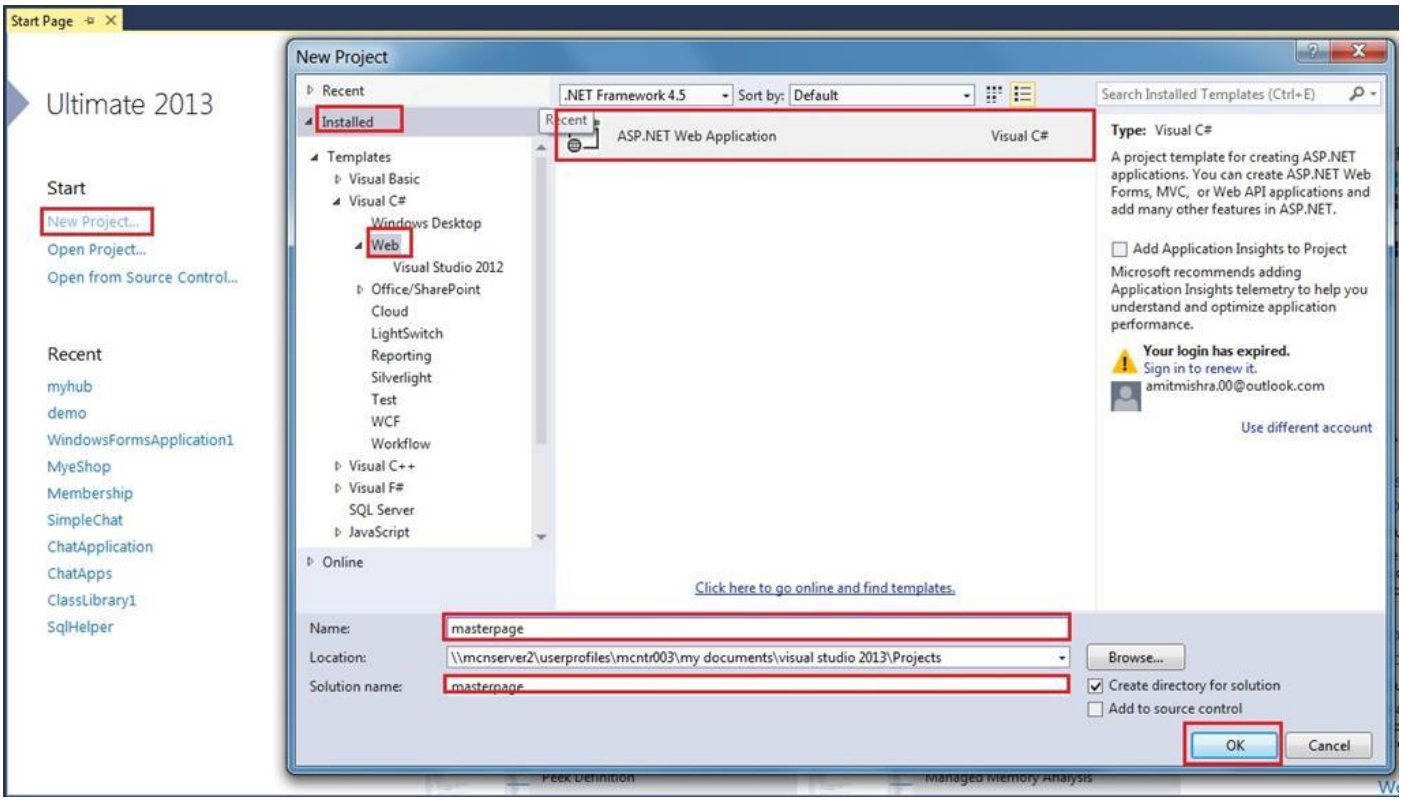
A **Master Page** is a nonprinting **page** that you can **use** as the template for the rest of the **pages** in your document. **Master pages** can contain text and graphic elements that will appear on all **pages** of a publication (i.e. headers, footers, **page** numbers, etc.)

Designing Website with Master Page, Theme and CSS

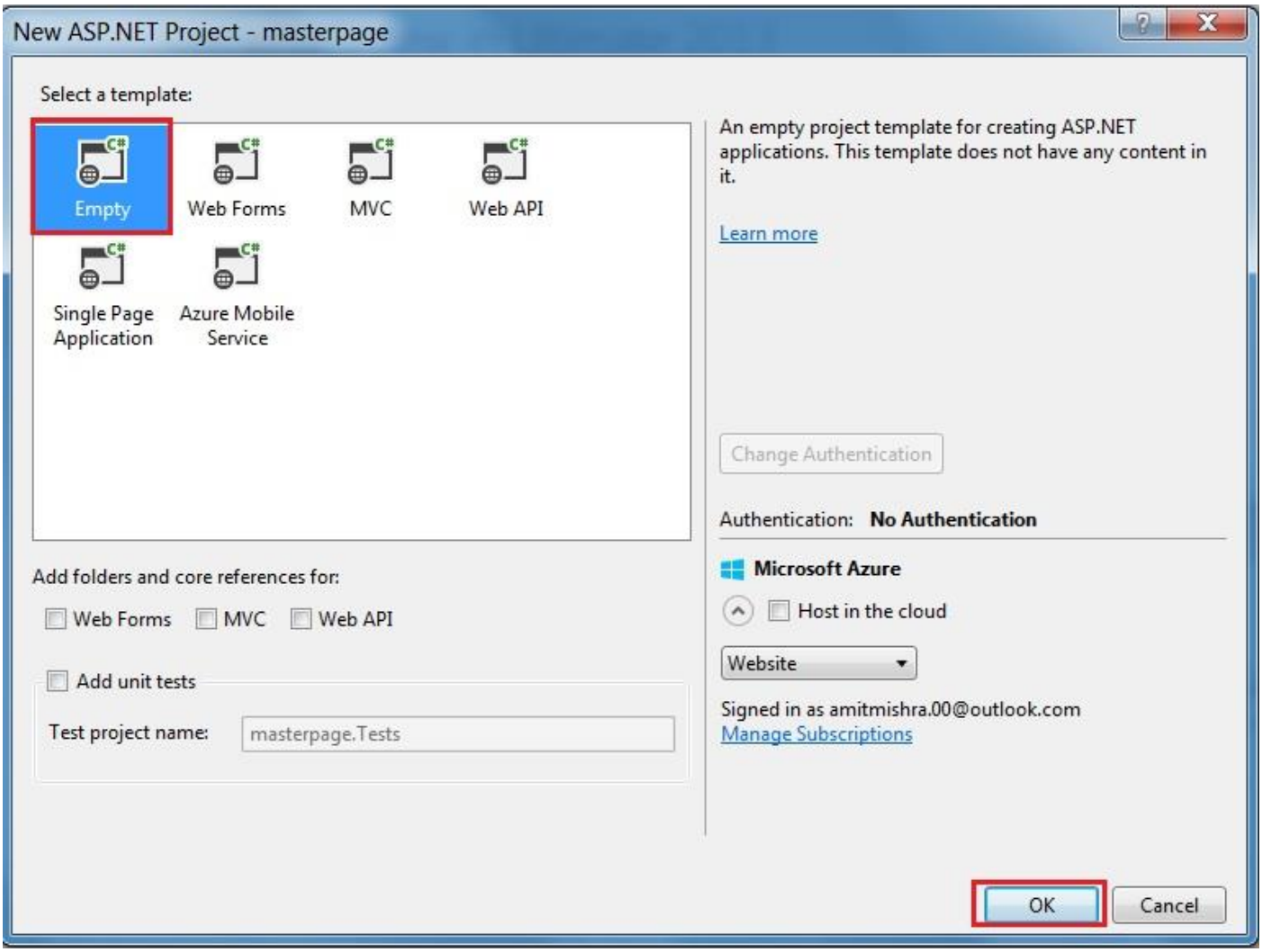
A master page provides the layout and functionality to other pages. Creating a master page in ASP.NET is very easy. Let's start creating master page step by step.

Step 1: Open new project in visual studio

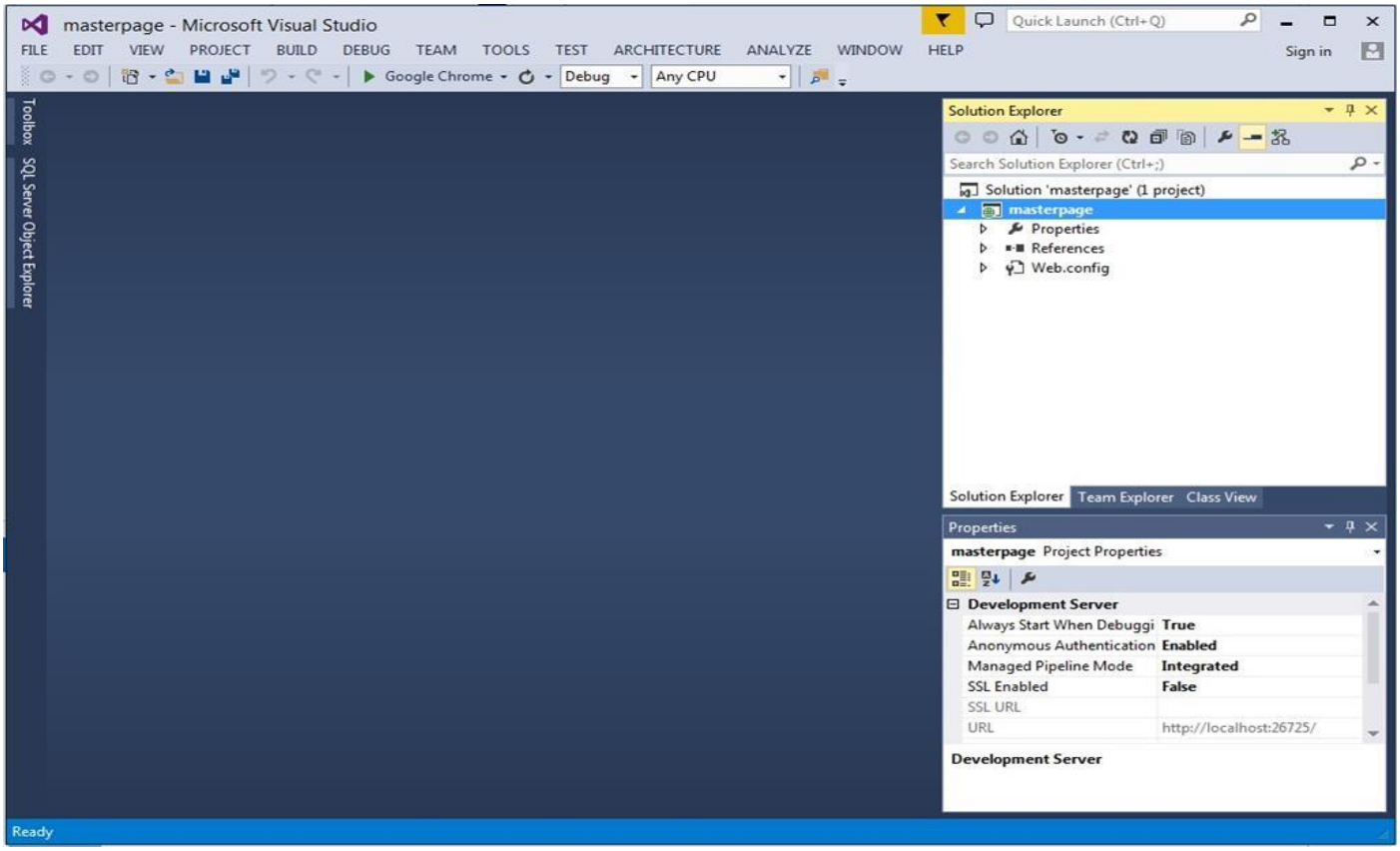
New project->Installed->Web->ASP.NET Web Application (shown in the picture),



After clicking OK button in the Window, select Empty (shown in the picture),



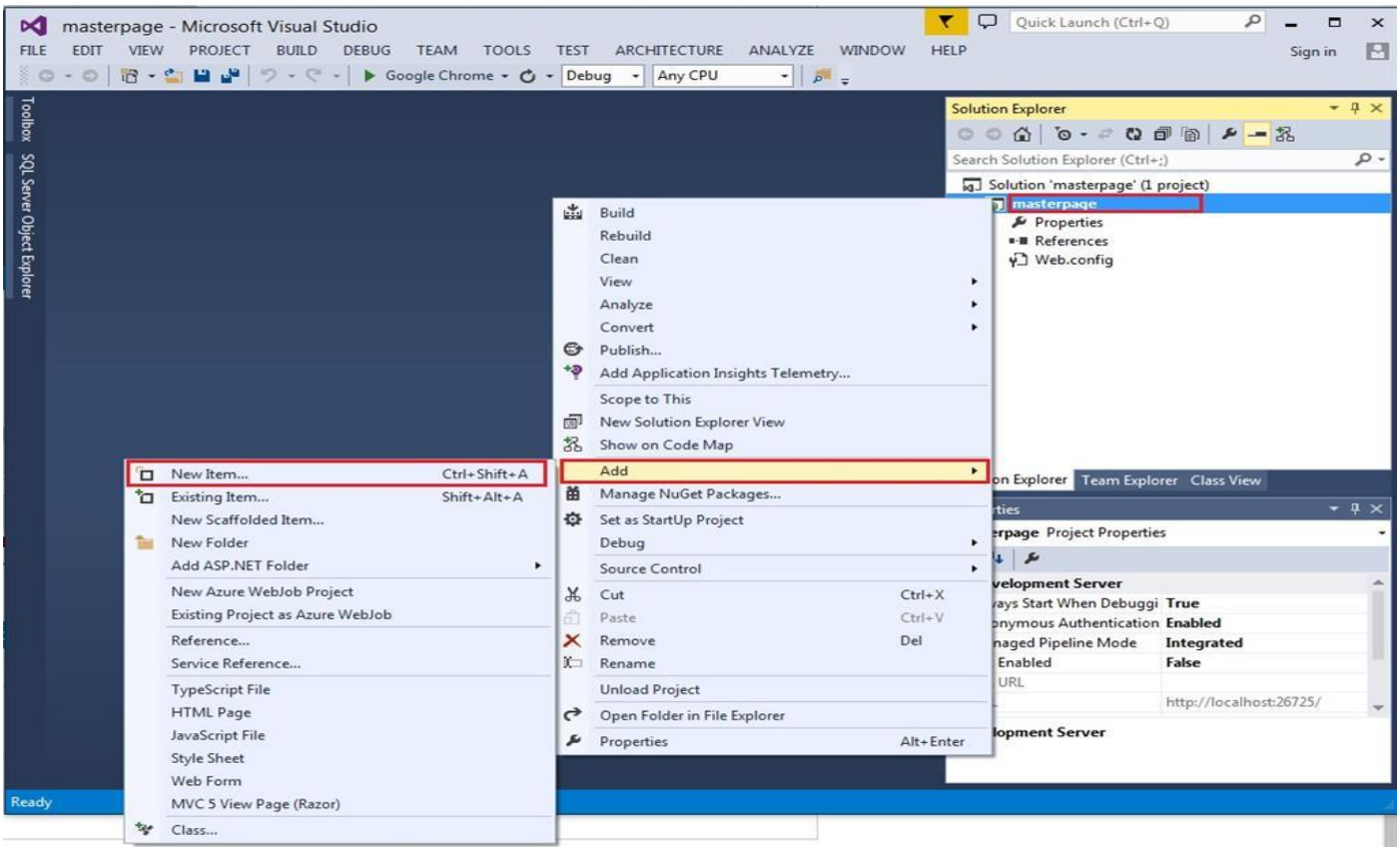
After clicking OK button, project "masterpage" opens but no file is there (shown in the picture),



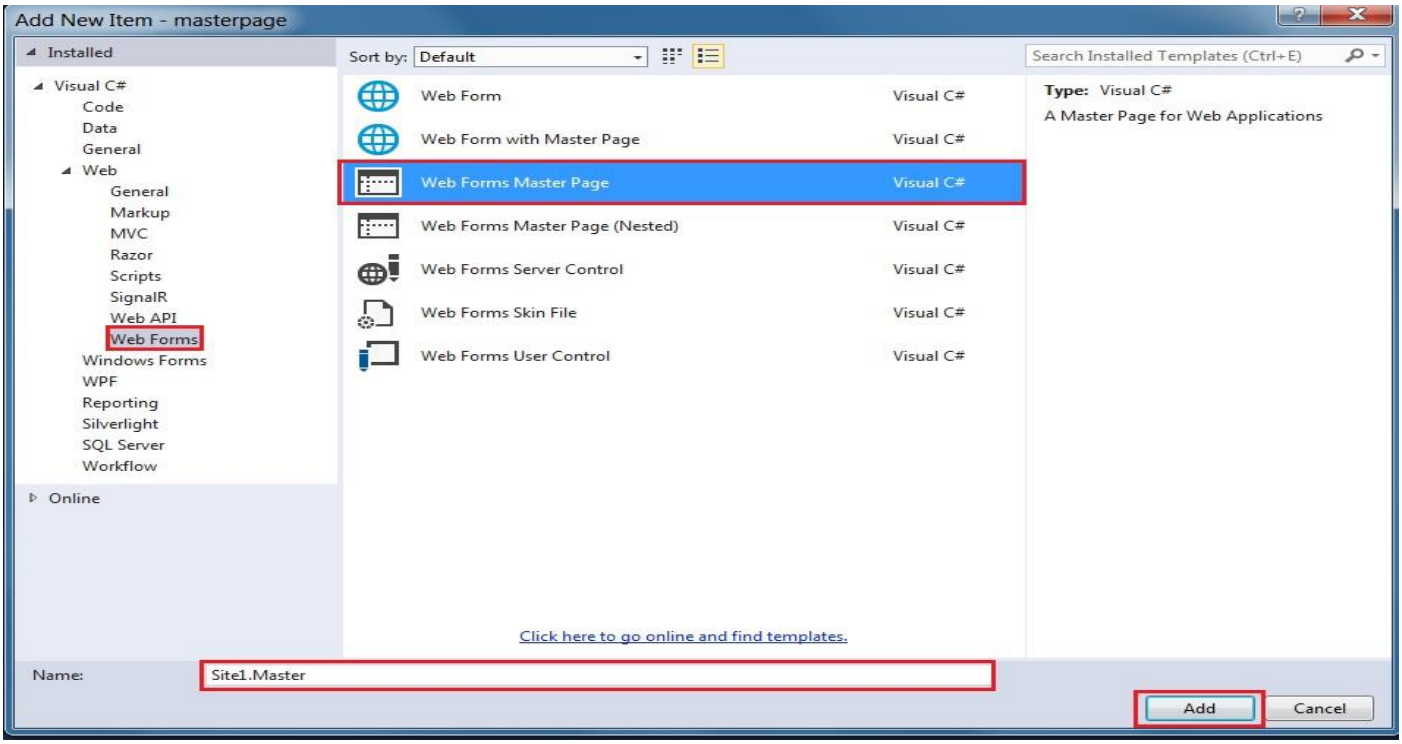
Step 2: Add new file in to our project.

Add the master page into our project.

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

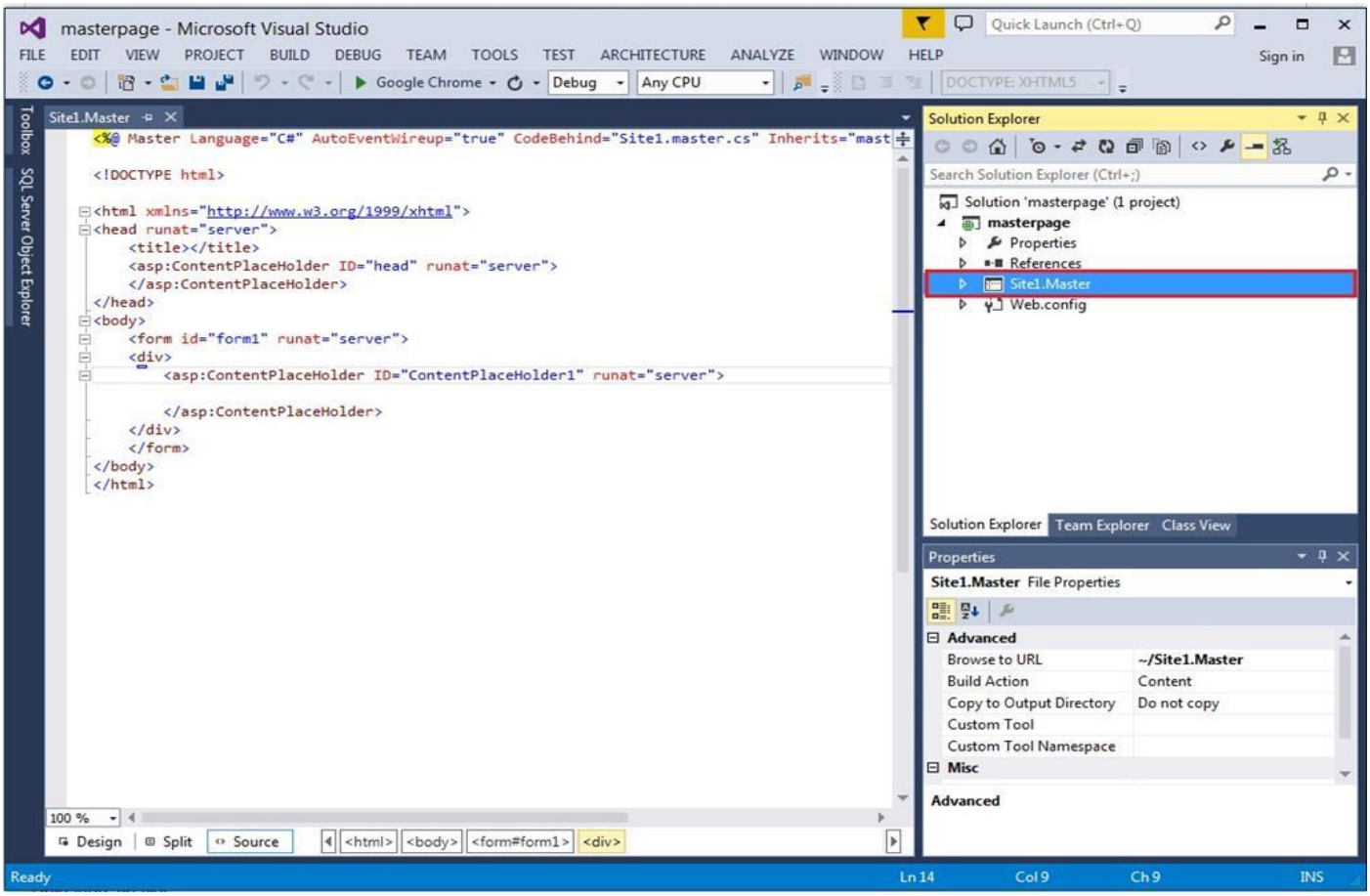


After clicking on new item, Window will open, select Web Form->Web Forms Master Page (shown in the picture),



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 3: Design the master page, using HTML.

HTML code of my master page is,

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site1.master.cs" Inherits="masterpage.Site1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>c# corner</title>
    <link href="css/my.css" rel="stylesheet" />
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <!DOCTYPE html>
<html>
<head>
    <title>my layout</title>
    <link rel="stylesheet" type="text/css" href="my.css">
</head>
<body>
<header id="header">
<h1>c# corner</h1>
</header>
<nav id="nav">
    <ul>
        <li><a href="home.aspx">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Article</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</nav>
<aside id="side">
    <h1>news</h1>
    <a href="#"><p>creating html website</p></a>
    <a href="#"><p>learn css</p></a>
    <a href="#">learn c#</a>
</aside>


    <div id="con">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>


<footer id="footer">
    copyright @c# corner
```

```
</footer>
</body>
</html>
    <form id="form1" runat="server">

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

CSS Code

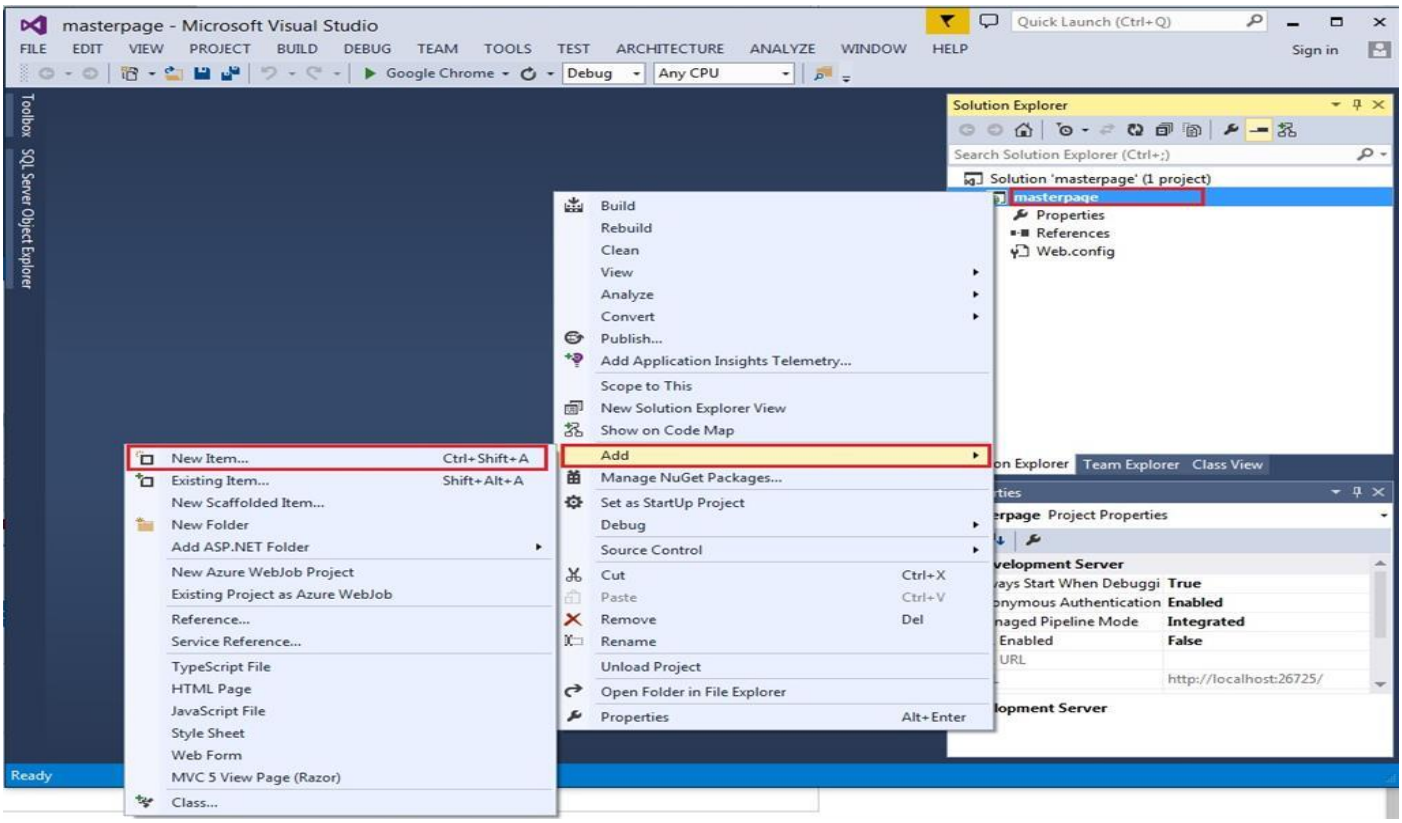
```
#header{
    color: #247BA0;
    text-align: center;
    font-size: 20px;
}
#nav{
    background-color:#FF1654;
    padding: 5px;
}
ul{

    list-style-type: none;
}
li a {
    color: #F1FAEE;
font-size: 30px;
column-width: 5%;
    }
    li
   {
    display: inline;
    padding-left: 2px;
    column-width: 20px;
   }
   a{
 text-decoration: none;
 margin-left:20px
    }
   li a:hover{
    background-color: #F3FFBD;
    color: #FF1654;
    padding:1%;
   }
   #side{
    text-align: center;
    float: right;
    width: 15%;
    padding-bottom: 79%;
    background-color: #F1FAEE;
   }
   #article{
```

```
    background-color: #EEF5DB;
    padding: 10px;
    padding-bottom: 75%;
  }
  #footer{
    background-color: #C7EFCF;
    text-align:center;
    padding-bottom: 5%;
    font-size: 20px;
} 
  #con{
      border:double;
      border-color:burlywood;
  }
```

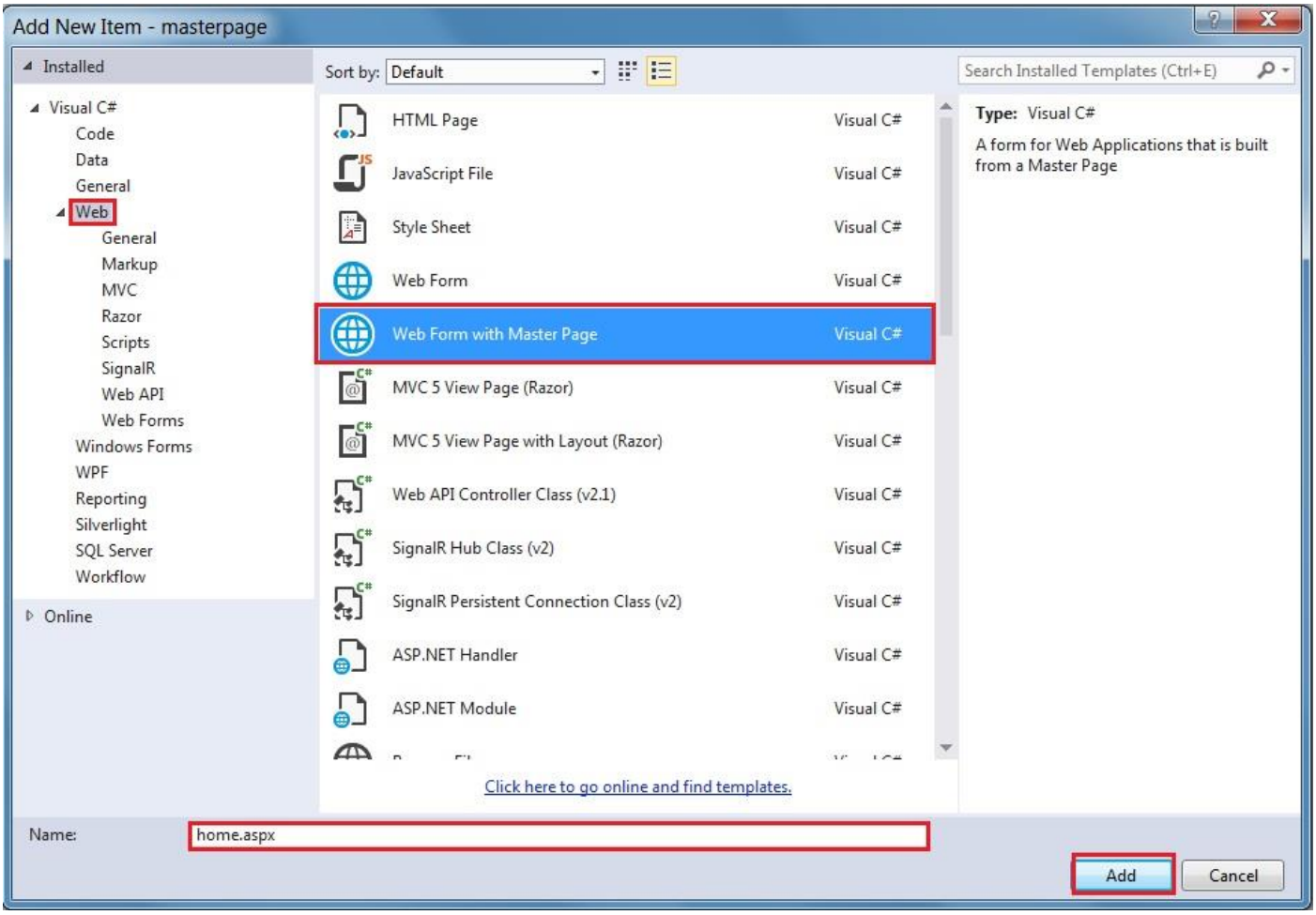
Our master page is designed. Move to the next step.

Step 4: Add web form in to our project.

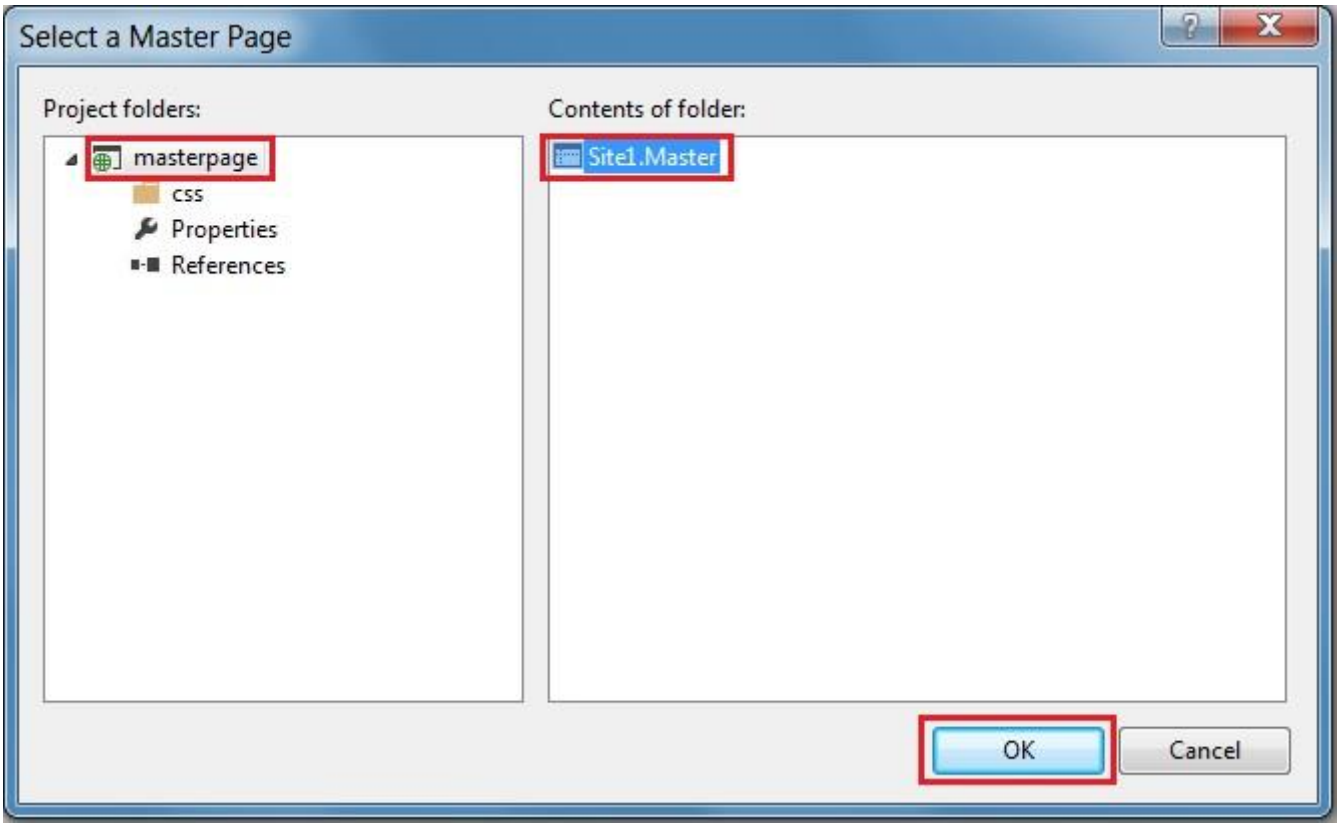
Right click on the project->Add->New item (shown in the picture),



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" AutoEventWireup="true" CodeBehind="home.aspx.cs" Inherits="masterpage.home" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <h1>Home page</h1>
</asp:Content>
```

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

Overview of CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions(CSS2). The main difference between css2 and css3 is follows −

- Media Queries
- Namespaces
- Selectors Level 3
- Color

CSS3 modules
CSS3 is collaboration of CSS2 specifications and new specifications, we can called this collaboration is **module**. Some of the modules are shown below −

- Selectors
- Box Model
- Backgrounds
- Image Values and Replaced Content
- Text Effects
- 2D Transformations
- 3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Page Output Caching

Output cache stores a copy of the finally rendered HTML pages or part of pages sent to the client. When the next client requests for this page, instead of regenerating the page, a cached copy of the page is sent, thus saving time.

Syntax for OutputCache directive:
<%@ OutputCache Duration="15" VaryByParam="None" %>

Put this directive under the page directive. This tells the environment to cache the page for 15 seconds. The following event handler for page load would help in testing that the page was really cached.

```
protected void Page_Load(object sender, EventArgs e)
{
  Thread.Sleep(10000);
  Response.Write("This page was generated and cache at:" +
  DateTime.Now.ToString());
}
```

Partial Page Caching

Page output caching caches the whole output of a page. Generally a web page contains static and dynamic contents. For example the data coming from database is not as frequently change as the advertisements changes. Therefore you would not wish to cache whole page because advertisements changes frequently.

Partial page caching enables you to cache portion of a page. It is also called as control caching or fragment caching.

Normally partial caching is performed using the **"User Control"**. You can do caching with the help of user control as same as page output caching but you have to do caching in user control.

First add a user control in your web project and write code as given below. In this application the user control name is TimeUserControl

TimeUserControl.ascx

```
<%@ OutputCache Duration="30" VaryByParam="none" %>
```

TimeUserControl.ascx.cs

```
using System;
public partial class TimeUserControl : System.Web.UI.UserControl
{
   protected void Page_Load(object sender, EventArgs e)
   {
      Label1.Text = "User Control Time </br> (Cached Time) =: " + DateTime.Now.ToString("hh:mm:ss");
   }
}
```

Use this user control on web page and do the following coding.

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "Normal Time =: " + DateTime.Now.ToString("hh:mm:ss");
}
```

The content of user control will be cached 30 seconds. If you refresh the page within 30 seconds, the user control time will not be change but the time that is available in web page is going to change every time when you postback the page or refresh.

Absolute Cache Expiration and Sliding Cache Expiration

Absolute and sliding expiration are two Time based expiration strategies.
Absolute Expiration: Cache in this case expires at a fixed specified date or time.

Example:
Cache. Insert("ABC", ds, null, DateTime.Now.AddMinutes(1), Cache.NoSlidingExpiration);

The cache is set to expire exactly two minutes after the user has retrieved the data.

Sliding Expiration: the cache duration increases in this case by the specified sliding expiration value every time the page is requested. More a page is requested it will remain in cache, whereas a less requested page will not remain in cache.

Example:
Cache.Insert("ABC", ds, null, Cache.NoAbsoluteExpiration, TimeSpan.FromMinutes(1));

Data Caching

Data caching means caching data from a data source. As long as the cache is not expired, a request for the data will be fulfilled from the cache. When the cache is expired, fresh data is obtained by the data source and the cache is refilled.

Example
To demonstrate data caching, create a new website and add a new web form on it. Add a SqlDataSource control with the database connection already used in the data access tutorials.
For this example, add a label to the page, which would show the response time for the page.
<asp:Label ID="lbltime" runat="server"></asp:Label>
Apart from the label, the content page is same as in the data access tutorial. Add an event handler for the page load event:

```
protected void Page_Load(object sender, EventArgs e)
{
   lbltime.Text = String.Format("Page posted at: {0}", DateTime.Now.ToLongTimeString());
}
```