**What are Master Pages ?**

**Master** **page**s let you make a consistent layout for your application, you can make one **master** **page** that holds the layout/look & feel and common functionality of your whole application and upon this **master** **page**, you can build all the other **page**s, we call these **page**s **Content Pages.** So simply you can build your **master** **page** and then build content **page**s, and while creating the content **page**s you bind them to the **master** **page** you have created before, those two **page**s are merged at runtime to give you the rendered **page**.

**Master Pages and ContentPlaceHolders**

The **master** **page** has the extension .**master** and it's actually an aspx **page** but with the directive <%@**Master** Language="vb"%>, instead of the standard **page** directive, almost the attributes of the **Master** directive are the same as that of the **page**, you can add any kind of controls the same as you design .aspx **page**s,

Every **MasterPage** should include at least one ContentPlaceHolder, this control is a placeholder for the content that will be merged at runtime, noting that the content **page** is just an aspx standard **page** but you should supply the attribute **MasterPage**File to the **page** directive to bind the content **page** to the **master** **page**, for example:

<%@ **Page** Language="VB" **MasterPage**File="~/**MasterPage**s/SiteLayout.**master**"%>

**Content Server Control**

Inside the content **page**s you will find one Content server control added by default, actually when you add controls you add them to the content server control. For example,

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"

Runat="Server">

The attribute ContentPlaceHolderID is very important as it decides what content will be bound to which ContentPlaceHolder on the **master** **page**, this is a very nice way to decide the location of where the contents will be displayed; so this way you can have multiple content on one content **page** and also multiple ContentPlaceHolders on the **master** **page**.

Note: The content **page**s don't include the common tags as <Body>, <Head>,etc. Remember that, that was the same with user controls, as after merging, there should be only one Body and Head tags.

Content Page

Master Page

Output Page

**Terminology**

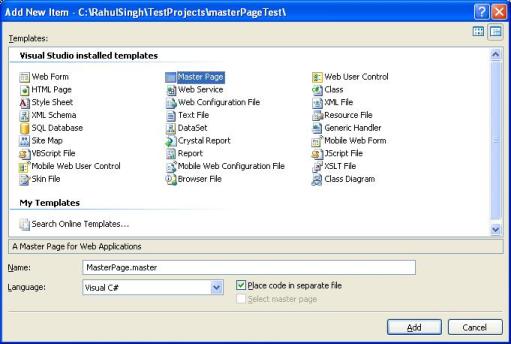
Let us look at the basic terminology that needs to be understood before jumping into master pages:

* 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 MasterPage 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 MasterPage's ContentPlaceHolder is located.

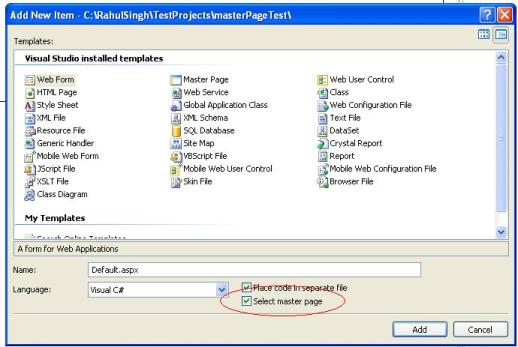
**Creating a MasterPage**

To create a master page, we need to:

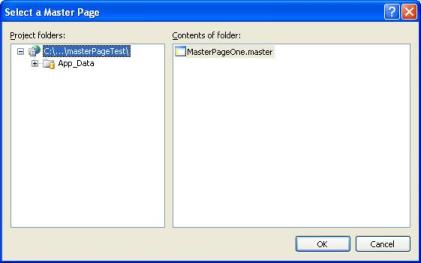
1. Go to "Add New Item".
2. Select the MasterPage.



1. Let's say our master page is MasterPageOne.Master.
2. We will now add a menu bar on this master page on top of the page. This Menu bar will be common to all the pages (since it is in Masterpage).
3. Once we have menubar added, we can have content pages use the master page.
4. Let's add few content pages like *default.aspx*, *about.aspx*, *Contact.aspx*. (We are simply creating some dummy pages with no functionality as we want to see the masterpage working, but these content pages can have any level of complex logic in them).
5. When we add these content pages, we need to remember to select the option of "Use master Page".



and select the master page.



Now let's look at the stuff that is important. When we look at the MasterPage, we will see that masterpage has a ContentPlaceHolder control. All the code that is common for the content pages is outside the ContentPlaceHolder control (in our case, a simple menubar).

**Adding the ContentPages**

If we look at our content pages, we will find a simple Content control added to each content page. This is the area where we will be adding our controls to be rendered along with the master page.

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

<h2>This is a the CONTACT page.</h2>

</asp:Content>

**Advantages of Master Page**

1. You can make updates in one place as they allow you to centralize the common functionality of your pages.   
2. With the help of Master pages, it is easy to create one set of controls and code and apply the results to a set of pages.   
For example, you can use controls on the master page to create a menu that applies to all pages.   
3. You can provide an object model which allows you to customize the master page from individual content pages.

**Nested Master Pages**

You can use more then one master page on your website. When more than one master page is used, you can make use of nested master pages.

For example, consider your company has a number of business partners or franchise companies. In such a scenario, you can define the layout and design for the standard elements such as logos, menus, copyright notices on the main master page of your company’s website. The franchise companies can then also define their own master pages and then nest their master page with the master page of your company.

**Understanding Nested Master Pages:**   
  
When a master page contains a reference of another master page, then it is called a **"nested master page"**. A single master page can have a reference of multiple master pages or a number of master pages can be componentized into a single master page. There is no limit to the number of child master pages in a project. The child masters can contain some unique properties of their own, besides using the layout and other properties of their parent master.