# MVC :: Creating Page Layouts with View Master Pages

In this tutorial, you learn how to create a common page layout for multiple pages in your application by taking advantage of view master pages. You can use a view master page, for example, to define a two-column page layout and use the two-column layout for all of the pages in your web application.

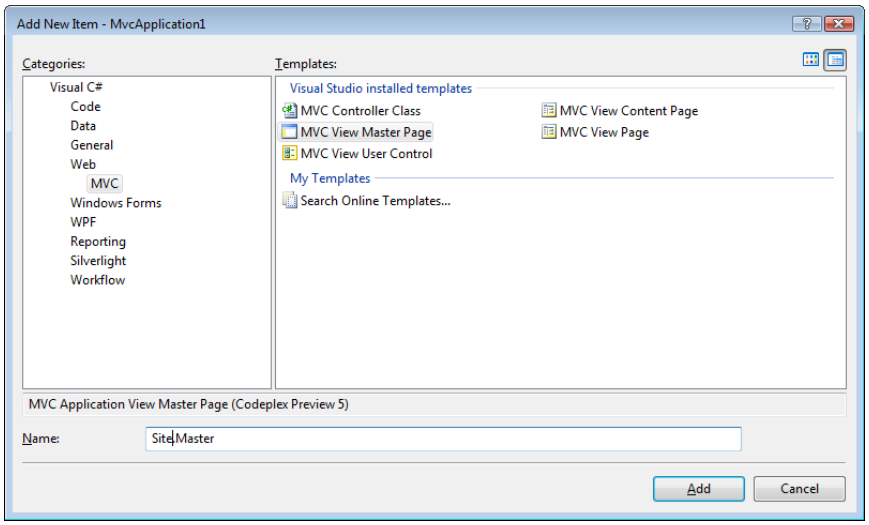
You also can take advantage of view master pages to share common content across multiple pages in your application. For example, you can place your website logo, navigation links, and banner advertisements in a view master page. That way, every page in your application would display this content automatically.

In this tutorial, you learn how to create a new view master page and create a new view content page based on the master page.

## Creating a View Master Page

Let's start by creating a view master page that defines a two-column layout. You add a new view master page to an MVC project by right-clicking the Views\Shared folder, selecting the menu option **Add, New Item**, and selecting the **MVC View Master Page** template (see Figure 1).

**Figure 1 – Adding a view master page**




You can create more than one view master page in an application. Each view master page can define a different page layout. For example, you might want certain pages to have a two-column layout and other pages to have a three-column layout.

A view master page looks very much like a standard ASP.NET MVC view. However, unlike a normal view, a view master page contains one or more <asp:ContentPlaceHolder> tags. The <ContentPlaceHolder> tags are used to mark the areas of the master page that can be overridden in an individual content page.

For example, the view master page in Listing 1 defines a two-column layout. It contains two <ContentPlaceHolder> tags. One <ContentPlaceHolder> for each column.

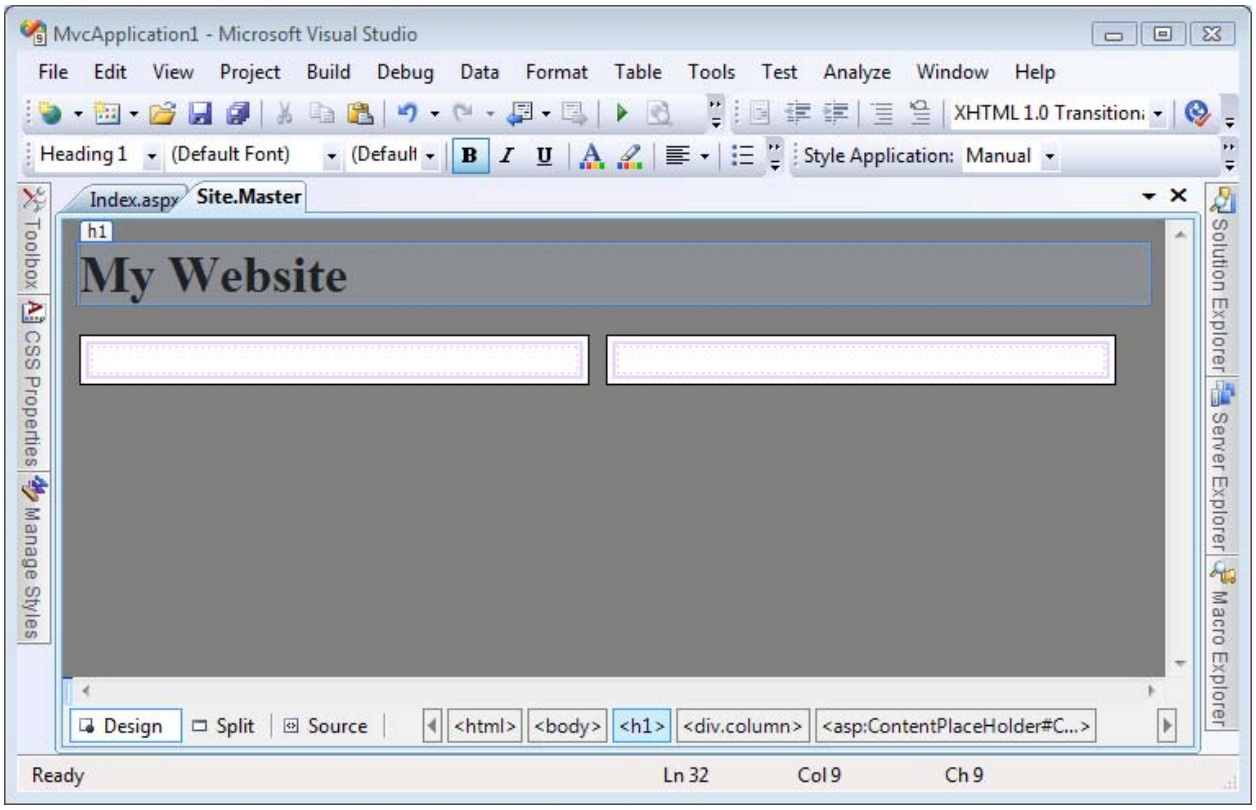
**Listing 1 – Views\Shared\Site.master**

```
<%@ Master Language="C#" AutoEventWireup="true"
    CodeBehind="Site.Master.cs"
    Inherits="MvcApplication1.Views.Shared.Main" %>

<!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>


    <style type="text/css">


    html

    {

        background-color:gray;

    }


    .column

    {

        float:left;

        width:300px;

        border:solid 1px black;

        margin-right:10px;

        padding:5px;

        background-color:white;

        min-height:500px;

    }


    </style>


    <asp:ContentPlaceHolder ID="head" runat="server">

    </asp:ContentPlaceHolder>

</head>

<body>
```

```
    <h1>My Website</h1>


    <div class="column">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
    runat="server">
        </asp:ContentPlaceHolder>
    </div>
    <div class="column">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
    runat="server">
        </asp:ContentPlaceHolder>
    </div>


</body>
</html>
```

The body of the view master page in Listing 1 contains two  tags that correspond to the two columns. The Cascading Style Sheet column class is applied to both  tags. This class is defined in the style sheet declared at the top of the master page. You can preview how the view master page will be rendered by switching to Design view. Click the Design tab at the bottom-left of the source code editor (see Figure 2).

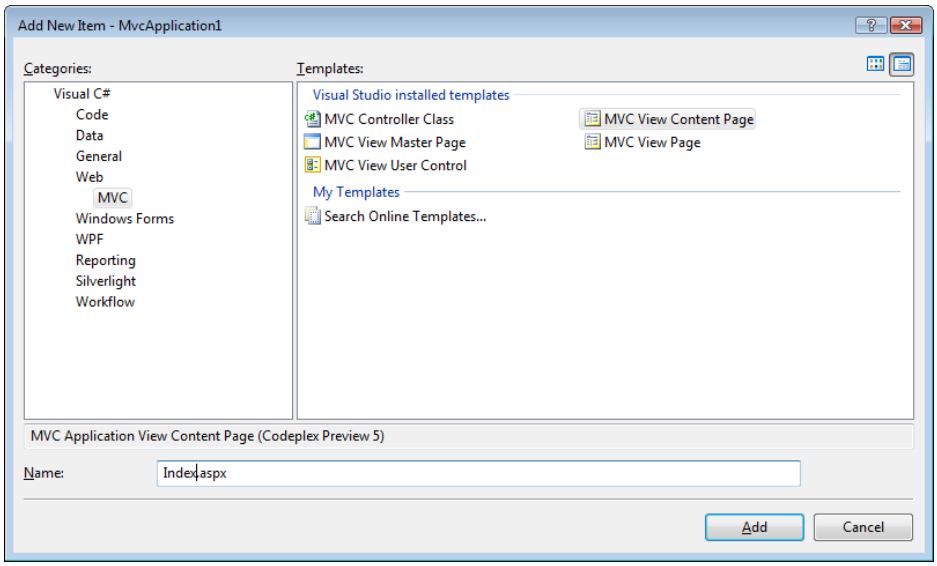
**Figure 2 – Previewing a master page in the designer**




# Creating a View Content Page

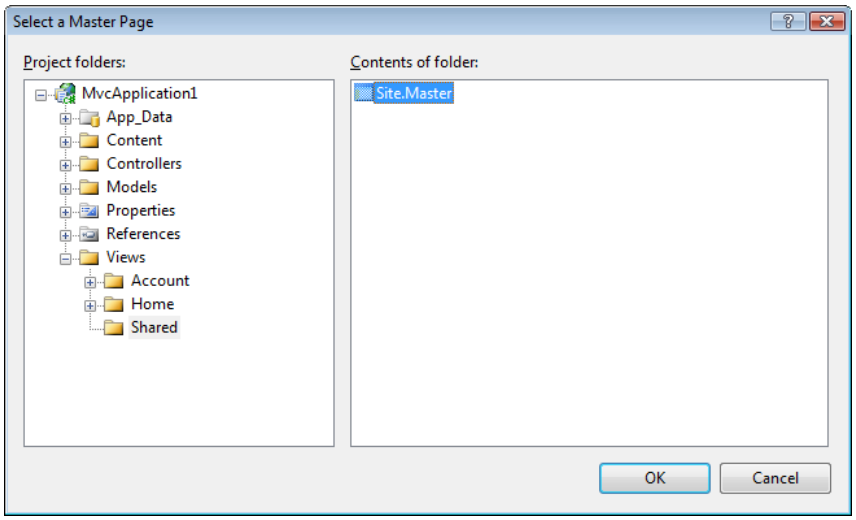
After you create a view master page, you can create one or more view content pages based on the view master page. For example, you can create an Index view content page for the Home controller by right-clicking the Views\Home folder, selecting **Add, New Item**, selecting the **MVC View Content Page** template, entering the name Index.aspx, and clicking the **Add** button (see Figure 3).

**Figure 3 – Adding a view content page**



After you click the Add button, a new dialog appears that enables you to select a view master page to associate with the view content page (see Figure 4). You can navigate to the Site.master view master page that we created in the previous section.

**Figure 4 – Selecting a master page**




After you create a new view content page based on the Site.master master page, you get the file in Listing 2.

**Listing 2 – Views\Home\Index.aspx**

```
<%@ Page Title="" Language="C#"
    MasterPageFile="~/Views/Shared/Site.Master"
    AutoEventWireup="true" CodeBehind="Index.aspx.cs"
    Inherits="MvcApplication1.Views.Home.Index" %>

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

</asp:Content>

<asp:Content ID="Content2"
    ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

</asp:Content>

<asp:Content ID="Content3"
    ContentPlaceHolderID="ContentPlaceHolder2" runat="server">

</asp:Content>
```

Notice that this view contains a <asp:Content> tag that corresponds to each of the <asp:ContentPlaceHolder> tags in the view master page. Each <asp:Content> tag includes a ContentPlaceHolderID attribute that points to the particular <asp:ContentPlaceHolder> that it overrides.

Notice, furthermore, that the content view page in Listing 2 does not contain any of the normal opening and closing HTML tags. For example, it does not contain the opening and closing <html> or <head> tags. All of the normal opening and closing tags are contained in the view master page.

Any content that you want to display in a view content page must be placed within a <asp:Content> tag. If you place any HTML or other content outside of these tags, then you will get an error when you attempt to view the page.

You don't need to override every <asp:ContentPlaceHolder> tag from a master page in a content view page. You only need to override a <asp:ContentPlaceHolder> tag when you want to replace the tag with particular content.

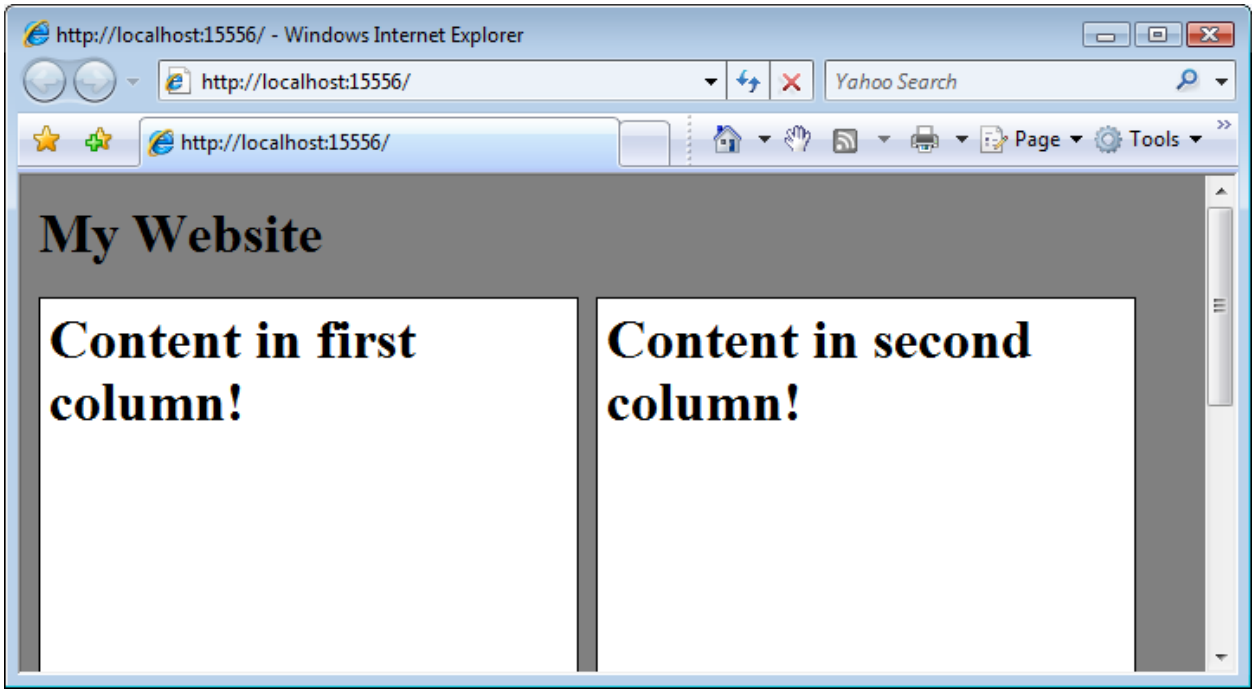
For example, the modified Index view in Listing 3 contains only two <asp:Content> tags. Each of the <asp:Content> tags includes some text.

**Listing 3 – Views\Home\Index.aspx (modified)**

```
<%@ Page Title="" Language="C#"
   MasterPageFile="~/Views/Shared/Site.Master"
   AutoEventWireup="true" CodeBehind="Index.aspx.cs"
   Inherits="MvcApplication1.Views.Home.Index" %>

<asp:Content ID="Content2"
   ContentPlaceHolderID="ContentPlaceHolder1" runat="server">


    <h1>Content in first column!</h1>




</asp:Content>

<asp:Content ID="Content3"
   ContentPlaceHolderID="ContentPlaceHolder2" runat="server">


    <h1>Content in second column!</h1>


</asp:Content>
```

When the view in Listing 3 is requested, it renders the page in Figure 5. Notice that the view renders a page with two columns. Notice, furthermore, that the content from the view content page is merged with the content from the view master page.

**Figure 5 – The Index view content page**



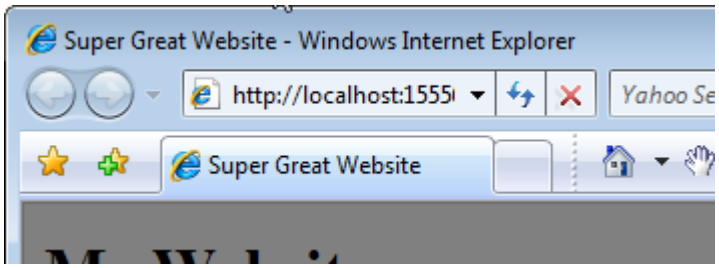

## Modifying View Master Page Content

One issue that you encounter almost immediately when working with view master pages is the problem of modifying view master page content when different view content pages are requested. For example, you want each page in your web application to have a unique title. However, the title is declared in the view master page and not in the view content page. So, how do you customize the page title for each view content page?

There are two ways that you can modify the title displayed by a view content page. First, you can assign a page title to the title attribute of the <%@ page %> directive declared at the top of a view content page. For example, if you want to assign the page title “Super Great Website” to the Index view, then you can include the following directive at the top of the Index view:

```
<%@ Page Title="Super Great Website" Language="C#"
MasterPageFile="~/Views/Shared/Site.Master" AutoEventWireup="true"
CodeBehind="Index.aspx.cs" Inherits="MvcApplication1.Views.Home.Index" %>
```

When the Index view is rendered to the browser, the desired title appears in the browser title bar:



There is one important requirement that a master view page must satisfy in order for the title attribute to work. The view master page must contain a <head runat=”server”> tag instead of a normal <head> tag for its header. If the <head> tag does not include the runat=”server” attribute then the title won’t appear. The default view master page includes the required <head runat=”server”> tag.

An alternative approach to modifying master page content from an individual view content page is to wrap the region that you want to modify in a <asp:ContentPlaceHolder> tag. For example, imagine that you want to change not only the title, but also the meta tags, rendered by a master view page. The master view page in Listing 4 contains a <asp:ContentPlaceHolder> tag within its <head> tag.

**Listing 4 – Views\Shared\Site2.master**

```
<%@ Master Language="C#" AutoEventWireup="true"
   CodeBehind="Site2.Master.cs"
   Inherits="MvcApplication1.Views.Shared.Site2" %>

<!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>

    <asp:ContentPlaceHolder ID="head" runat="server">

        <title>Please change my title</title>

        <meta name="description" content="Please provide a
   description" />

        <meta name="keywords" content="keyword1,keyword2" />

    </asp:ContentPlaceHolder>

</head>

<body>

    <div>

        <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
   runat="server">


        </asp:ContentPlaceHolder>

    </div>

</body>

</html>
```

Notice that the <asp:ContentPlaceHolder> tag in Listing 4 includes default content: a default title and default meta tags. If you don’t override this <asp:ContentPlaceHolder> tag in an individual view content page, then the default content will be displayed.

The content view page in Listing 5 overrides the <asp:ContentPlaceHolder> tag in order to display a custom title and custom meta tags.

**Listing 5 – Views\Home\Index2.aspx**

```
<%@ Page Title="" Language="C#"
   MasterPageFile="~/Views/Shared/Site2.Master"
   AutoEventWireup="true" CodeBehind="Index2.aspx.cs"
   Inherits="MvcApplication1.Views.Home.Index2" %>

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

    <title>The Index2 Page</title>

    <meta name="description" content="Description of Index2 page" />

    <meta name="keywords" content="asp.net,mvc,cool,groovy" />

</asp:Content>

<asp:Content ID="Content2"
   ContentPlaceHolderID="ContentPlaceHolder1" runat="server">


    Just some content in the body of the page.


</asp:Content>
```

# Summary

This tutorial provided you with a basic introduction to view master pages and view content pages. You learned how to create new view master pages and create view content pages based on them. We also examined how you can modify the content of a view master page from a particular view content page.