murach's
ASP.NET 2.0
upgrader's guide
VB EDITION

(Chapter 3)

Thanks for downloading this chapter from *Murach's ASP.NET 2.0 Upgrader's Guide: VB Edition*. We hope it will show you how easy it is to learn from any Murach book, with its paired-pages presentation, its "how-to" headings, its practical coding examples, and its clear, concise style.

To view the full table of contents for this book, you can go to our **web site**. From there, you can read more about this book, you can find out about any additional downloads that are available, and you can review our other books on .NET development.

Thanks for your interest in our books!

MIKE MURACH & ASSOCIATES, INC.

1-800-221-5528 • (559) 440-9071 • Fax: (559) 440-0963
murachbooks@murach.com • **www.murach.com**

Books for .NET programmers

Murach's ASP.NET 2.0 Upgrader's Guide: C# Edition (ISBN 1-890774-35-9)
Murach's ASP.NET 2.0 Upgrader's Guide: VB Edition (ISBN 1-890774-36-7)

Murach's C# (ISBN 1-890774-22-7)
Murach's Beginning Visual Basic .NET (ISBN 1-890774-15-4)
Murach's ASP.NET Web Programming with VB.NET (ISBN 1-890774-20-0)
Murach's VB.NET Database Programming with ADO.NET (ISBN 1-890774-19-7)
Murach's SQL for SQL Server (ISBN 1-890774-16-2)

Please check www.murach.com
for other books on .NET 2.0 subjects

3

How to use master pages

A master page makes it easy for you to create pages that have common elements such as banners and navigation menus. That's why it is one of the most important new features of ASP.NET 2.0. In fact, you may decide that you're going to use one master page for every group of pages that you develop.

How to create master pages

A *master page* is a page that provides a framework within which the content from other pages can be displayed. Master pages make it easy to include banners, navigation menus, and other elements on all of the pages in an application. In the topics that follow, you'll learn how to create master pages in your ASP.NET applications.

An introduction to master pages

Figure 3-1 shows the basics of how master pages work. As you can see, the page that's actually sent to the browser is created by combining elements from a master page and a *content page*. The content page provides the content that's unique to each page in the application, while the master page provides the elements that are common to all pages. In this example, the master page (MasterPage.master) provides a banner at the top of each page, a simple navigation menu at the side of each page, and a message that indicates how many days remain until Halloween at the bottom of each page.

In addition, the master page contains a *content placeholder* that indicates where the content from each content page should be displayed. In this example, the content page is the Order.aspx page, and its content is displayed in the content placeholder in the central portion of the master page.

Notice that the name of the content page is Order.aspx, the same as the Order page that you saw in chapter 2. In other words, when you use master pages, the individual pages of your web application become the content pages. You'll learn how to create content pages or convert existing ASP.NET pages to content pages in figure 3-6.

The Cart application with a master page

Master page (MasterPage.master)
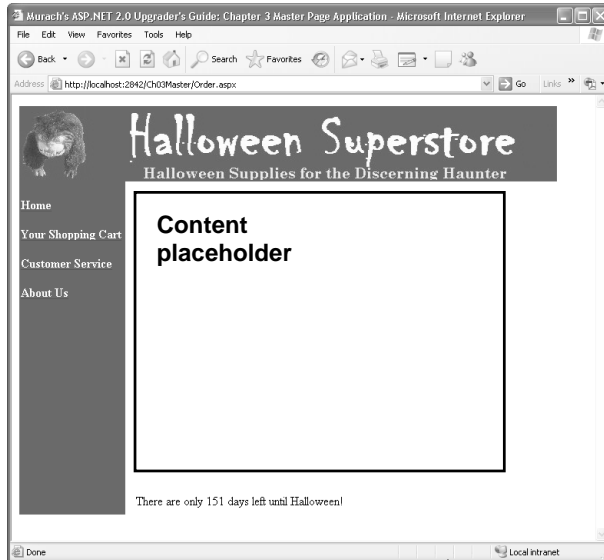
Content page (Order.aspx)
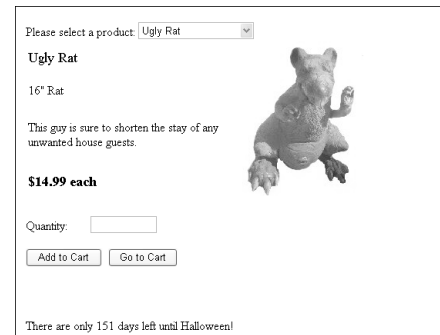
Rendered page
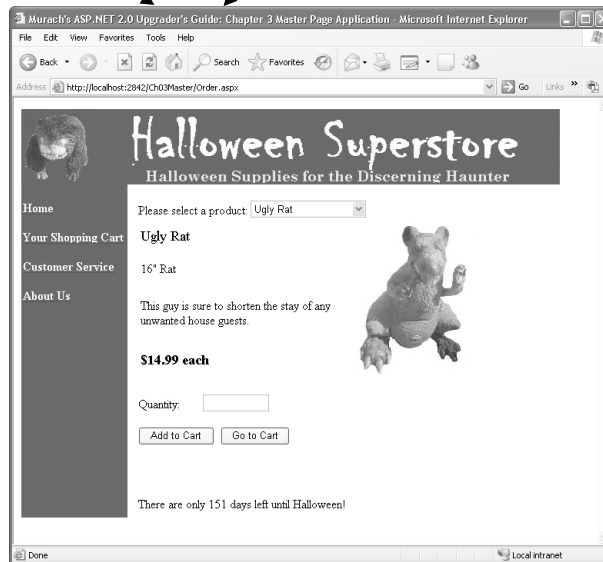
Description

- A *master page* provides a framework in which the content of each page on a web site is presented. Master pages make it easy to create pages that have a consistent look.

- The pages that provide the content that's displayed in a master page are called *content pages*.

- The content of each content page is displayed in the master page's *content placeholder*.

Figure 3-1   An application that uses a master page

How to create a master page

As figure 3-2 shows, you create a master page by using the Website→Add New Item command. Master Page is listed as one of the templates in the Add New Item dialog box. Select this template, select Visual Basic as the language, and click Add to create the master page. The default name for a master page is MasterPage.master.
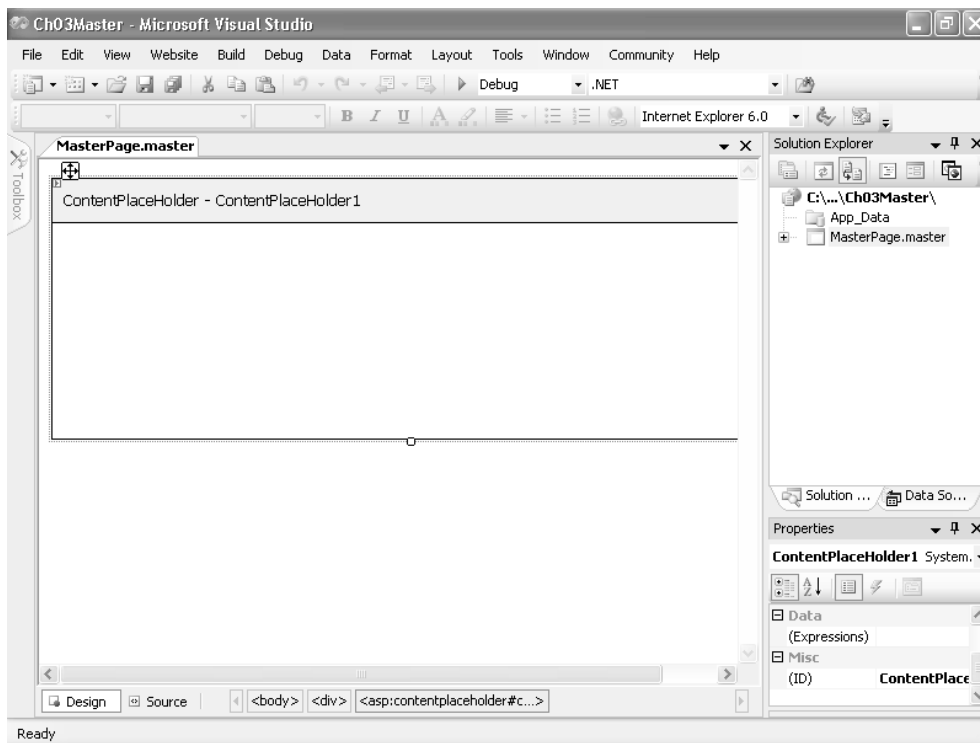
The master page created from the template includes a ContentPlaceHolder control that will contain the content page, but nothing else. You'll learn more about the ContentPlaceHolder control in the next figure. For now, just realize that it marks the location on the rendered page where the content from the content page will be displayed.

You can develop the master page by adding elements outside of the ContentPlaceHolder control. For example, to create the master page in figure 3-1, you add an image for a banner above the placeholder, navigation links to the left of the placeholder, and a label to display the days remaining until Halloween below the placeholder. Typically, you'll use an HTML table to specify the layout of these elements.

Note that an application can contain more than one master page. This allows you to create an application that has two or more sections with distinct page layouts. For example, you may want to use one master page for all of the content pages in the online shopping section of a web site, and another master page for the content pages in the customer service section.

In addition, you should realize that a master page can have more than one content placeholder. This lets you create a page layout that has custom content in two or more different areas of the page. To create an additional content placeholder, you simply drag the ContentPlaceHolder control from the Standard tab of the Toolbox onto the master page and give it a unique ID.

A new master page in Design view

Description

- To add a master page to a project, choose the Website➔Add New Item command. Then, in the Add New Item dialog box, select Master Page from the list of templates, specify the name of the master page you want to create in the Name text box (the default is Master Page.master), and select the programming language. Then, click Add.

- The content placeholder appears as a control in the Web Forms Designer. Although you can change the position of the content placeholder, you can't edit its contents from the master page. Instead, you add content to the master page by creating content pages as described later in this chapter.

- Any elements you add to the master page outside of the content placeholder will appear on every content page that uses the master page.

- Although most master pages have just one content placeholder, you can create more than one content placeholder if you need to. In that case, each placeholder displays a portion of the content of each content page.

- An application can have more than one master page, and each content page specifies which master page should be used to display the content page.

- The aspx file for a master page uses the extension .master. The code-behind file uses .master.vb.

Figure 3-2   How to create a master page

The aspx code for a new master page

The listing at the top of figure 3-3 shows the aspx code that's generated when you create a master page using the Master Page template. As you can see, this code is similar to the aspx code generated for a regular ASP.NET web page, with two important differences.

First, instead of a Page directive, the code begins with a Master directive. This indicates that the file contains a master page rather than a regular ASP.NET page. Second, the Div element that normally contains the content for the page now contains a ContentPlaceHolder control.

Notice that the master page file is itself a well-formed HTML document with html, head, and body elements. The body element includes a form element, which in turn contains the ContentPlaceHolder control. Any elements you add to the master page should appear within the form element, but outside of the ContentPlaceHolder control.

The aspx code for a new master page

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb" Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
        </asp:contentplaceholder>
    </div>
    </form>
</body>
</html>
```

Attributes of the Master page directive

Attribute	Description
Language	Specifies the language used for any code required by the page.
CodeFile	Specifies the name of the code-behind file.
Inherits	Specifies the name of the page class defined in the code-behind file.

Attributes of the ContentPlaceHolder control

Attribute	Description
ID	Specifies the name of the content placeholder.
Runat	Specifies that the control is a server-side control.

Description

- A master page must begin with a Master page directive and should include at least one ContentPlaceHolder control.
- Any HTML or aspx elements that you add to the master page will be displayed on every page that uses the master page along with the ContentPlaceHolder control.

Figure 3-3   The aspx code for a new master page

The aspx code for the Halloween Store master page

Figure 3-4 shows the complete aspx code for the master page in figure 3-1. Although this listing fills the entire page, there's nothing complex about it. So you shouldn't have any trouble understanding its elements.

The banner at the top of the page is displayed using an image control. After the banner, a table element controls the layout of the rest of the page. The first row of this table specifies a height of 400 pixels. It has three cells. The first cell contains a simple navigation menu built using <a> tags. The background color for this cell is set to red. The second cell is a small (10 pixel) spacer cell that gives some space between the navigation menu and the content. And the third cell contains the content placeholder.

The second row defines the footer that appears at the bottom of the page. It also has three cells, each the same width as the cells in the first row. However, the height of this row is set to 25 pixels. The third cell in this row contains a label control named lblMessage. This label will be used to display the number of days that remain until Halloween.

The aspx code for the master page

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb" Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Murach's ASP.NET 2.0 Upgrader's Guide: Chapter 3 Master Page
Application</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/banner.jpg" /><br />
    <table cellpadding="2" cellspacing="0">
      <tr height="400">
        <td style="width: 157px" valign="top" bordercolor="red" bgcolor="red">
            <br />
            <a href="Order.aspx">
                <span style="color: #ffffff"><b>Home</b></span>
            </a><br /><br />
            <a href="Cart.aspx">
                <span style="color: #ffffff"><b>Your Shopping Cart</b></span>
            </a><br /><br />
            <a href="Service.aspx">
                <span style="color: #ffffff"><b>Customer Service</b></span>
            </a><br /><br />
            <a href="About.aspx">
                <span style="color: #ffffff"><b>About Us</b></span>
            </a></td>
        <td style="width: 10px"></td>
        <td style="width: 704px" valign="top">
            <asp:contentplaceholder id="Main" runat="server">
            </asp:contentplaceholder>
        </td>
      </tr>
      <tr height="25">
        <td bgcolor="red" bordercolor="red"
            style="width: 153px" valign="top"></td>
        <td style="width: 10px"></td>
        <td style="width: 704px" valign="top">
            <asp:Label ID="lblMessage" runat="server"></asp:Label>
        </td>
      </tr>
    </table>
    </form>
</body>
</html>
```

Description

- Most master pages include elements like banners and navigation controls.
- It's common to use tables to provide the layout for the elements on the master page, including the content placeholder.

Figure 3-4   The aspx code for the Halloween Store master page

The code-behind file for the master page

Master pages have events just like regular ASP.NET pages. So it's important to realize that most of these events are raised *after* the corresponding events for the content page are raised. For example, the Page Load event for the master page will be processed after the Page Load event for the content page. Likewise, any control events for the content page are processed before any control events for the master page. Note, however, that both the content page and the master page Load events are processed before any of the control events are processed.

Figure 3-5 shows the code-behind file for the master page in figure 3-4. This code-behind file includes a Page_Load procedure that's executed when the master page loads. As you can see, this procedure calls a procedure named DaysUntilHalloween, which calculates and returns the number of days remaining until October 31. Then, an appropriate message is assigned to the Text property of the lblMessage label.

The code behind file for the master page

```
Partial Class MasterPage
    Inherits System.Web.UI.MasterPage

    Protected Sub Page_Load(ByVal sender As Object, _
            ByVal e As System.EventArgs) Handles Me.Load
        Dim iDaysUntil As Integer = DaysUntilHalloween()
        If iDaysUntil = 0 Then
            lblMessage.Text = "Happy Halloween!"
        ElseIf iDaysUntil = 1 Then
            lblMessage.Text = "Tomorrow is Halloween!"
        Else
            lblMessage.Text = "There are only " & iDaysUntil _
                & " days left until Halloween!"
        End If
    End Sub

    Private Function DaysUntilHalloween() As Integer
        Dim dtmHalloween As Date = New DateTime(DateTime.Today.Year, 10, 31)
        If DateTime.Today > dtmHalloween Then
            dtmHalloween.AddYears(1)
        End If
        Dim tsTimeUntil As TimeSpan = dtmHalloween - DateTime.Today
        Return tsTimeUntil.Days
    End Function

End Class
```

Description

- Master pages have events just like regular ASP.NET pages. For this master page, the Page Load event is used to display the number of days remaining until Halloween.

- Most events for the content page are raised before the corresponding events for the master page. For example, the Page Load event for the content page is raised before the Page Load event for the master page. Similarly, events for controls in the content page are raised before events for controls in the master page.

Figure 3-5   The code-behind file for the master page

How to create and develop content pages

Once you create a master page, you can create and develop the content pages for the master page. The topics that follow show how.

How to create a content page

Figure 3-6 shows how to create a content page. In short, you use the same procedure to create a content page that you use to create a regular page, but you check the Select a Master Page check box. Then, you can choose the master page you want to use for the content page from the Select a Master Page dialog box that's displayed.

Alternatively, you can select the master page you want to use in the Solution Explorer. Then, choose the Website➔Add Content Page command. This creates a content page for the selected master page. Note that when you use this technique, the content page is automatically named Default.

The code example in this figure shows the code that's generated when you create a new content page for the master page shown in figure 3-4. This code is quite different from the code that's generated when you create a regular ASP.NET page. Although the Page directive includes the same information as a regular ASP.NET page, it also includes a MasterPageFile attribute that specifies the master page you selected. And the rest of the content page is completely different from a normal ASP.NET page.

Before I describe the other differences, you should know that the title you specify in the Title attribute of the Page directive of a content page overrides any title you specify in the master page. That way, you can display a different title for each content page. If you want to use the same title for each content page, however, you can specify the title in the master page and then delete the Title attribute from the content pages.

Unlike normal ASP.NET pages, content pages don't include a Doctype directive or any structural HTML elements such as html, head, body, or form. That's because those elements are provided by the master page. Instead, the content page includes an ASP.NET Content element that indicates which content placeholder the page should be displayed in. Then, you place the content that you want to display on the page between the start and end tags of this element.

This figure also includes a procedure for converting a regular page to a content page. You'll need to follow this procedure if you start a web site without using master pages, and later decide to use master pages. Unfortunately, though, Visual Studio doesn't provide a way to automatically do this. As a result, you'll have to manually edit each of the pages to add the MasterPageFile attribute to the Page directive, remove the Doctype directive and structural HTML elements (html, head, body, and form), and add a Content element.

The aspx code for a new page that uses the master page in figure 3-4

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
    AutoEventWireup="false" CodeFile="Order.aspx.vb"
    Inherits="Order" Title="Untitled Page" %>

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

How to create a new content page

- One way is to choose the Website➜Add New Item command. Then, select Web Form from the list of templates, enter the name for the form, check the Select a Master Page check box, and click Add. When the Select a Master Page dialog box appears, select the master page you want and click OK.
- Another way is to select the master page in the Solution Explorer, then choose the Website➜Add Content Page command.

How to convert a regular ASP.NET page to a content page

- First, add a MasterPageFile attribute to the Page directive that specifies the URL of the master page. Next, replace the Div element that contains the actual content of the page with a Content element as shown above. Then, delete everything that's outside this Content element except for the Page directive.

Two other ways to specify the master page

In the web.config file

```
<system.web>
    .
    <pages masterPageFile="MasterPage.master" />
    .
</system.web>
```

In the Page_PreInit procedure

```
Protected Sub Page_PreInit(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.PreInit
    MasterPageFile = "MasterPage.master"
End Sub
```

Description

- The Page directive in the aspx code for a content page includes a MasterPageFile attribute that specifies the name of the master page.
- The aspx code for a content page includes a Content element that indicates the ID of the content placeholder where the content for the page should be displayed. Any content you create for the page should be placed between the start and end tags for this element.
- You can also specify the master page in the web.config file or in the Page_PreInit procedure. However, the Web Forms Designer doesn't support either of these techniques, so you won't be able to view the content page in Design view.

Figure 3-6   How to create a content page

Because this conversion procedure is error prone, it pays to use master pages for all but the simplest of applications, even if each master page contains only a content placeholder. Then, when you're ready to provide a consistent look to the pages within the application, you can enhance the master pages.

This figure also shows two other ways to specify which master page is used with a content page. First, you can add a <pages> element to the web.config file with a MasterPageFile attribute that specifies the master page to be used with all pages that don't specify a master file. Second, you can specify the master page at runtime by setting the MasterPageFile attribute of the page in the Page_PreInit procedure. Note, however, that the Web Forms Designer doesn't support either of these techniques. If you use them, then, you won't be able to view or edit your content pages in Design view.

How to add content to a page

Figure 3-7 shows how a content page appears in Design view. As you can see, the master page is displayed, but it is dimmed, and you can't edit any of the master page elements from this view. However, you can click in the Content control, and then edit the content of the page by adding text or dragging controls from the Toolbox. Later, when you switch to Source view, any elements that you've added will appear between the Content element's start and end tags.

If you work with a master page that has more than one content placeholder, there will be a separate Content control for each placeholder. Then, you can edit the contents for each of those controls.

A content page in Design View

Description

- When you display a content page in Design view, the elements from the master page are dimly displayed so you can see how they will affect the final appearance of the page.
- To add the content for a page, click the content placeholder. Then, you can type text or use the toolbox to drag controls into the content area of the page. Any text or other elements you add will be placed between the start and end tags of the Content element in the aspx file.

Note

- If you can't edit the contents of the placeholder, click the Smart Tag icon in the upper-right corner of the placeholder and choose Create Custom Content from the menu that appears. This is sometimes necessary, probably due to a bug in Visual Studio.

Figure 3-7   How to add content to a page

How to access master page controls from a content page

In many applications, you need to access one or more of the controls in a master page from one of the application's content pages. For example, the master page shown earlier in this chapter has a label in the footer area that normally displays the number of days remaining until Halloween. But what if you want to display other information in this label when certain content pages are displayed?

For example, when the user is shopping for products with the Order.aspx page, you may want to display the number of items currently in the shopping cart instead of the number of days left until Halloween. To do that, you can expose a master page control as a public property, and then access the property from the content page.

How to expose a master page control as a public property

The easiest way to access a control on a master page from a content page is to create a public property that provides access to the control. Figure 3-8 illustrates a code-behind file for a master page that shows you how to do that.

First, you create a public property in the master page that identifies the control you want to be able to access. In this case, the property is a Label type, and the property is named MessageLabel. Then, you code get and set procedures for the property. Here, the get procedure returns the lblMessage label, and the set procedure assigns the property value to lblMessage.

Please notice that I also added another If statement to the Page_Load procedure for the master page. Now, the lblMessage label is set to the number of days left until Halloween only if the value of the label's Text property is empty. That way, if the content page has assigned a value to this label in its Page_Load procedure, the master page's Page_Load procedure won't overwrite the value. This works because the content page's Page_Load procedure is called before the master page's Page_Load procedure.

The code-behind file for a master page that provides a public property

```
Partial Class MasterPage
    Inherits System.Web.UI.MasterPage

    Public Property MessageLabel() As Label
        Get
            Return lblMessage
        End Get
        Set(ByVal value As Label)
            lblMessage = value
        End Set
    End Property

    Protected Sub Page_Load(ByVal sender As Object, _
            ByVal e As System.EventArgs) Handles Me.Load
        If lblMessage.Text = "" Then
            Dim iDaysUntil As Integer = DaysUntilHalloween()
            If iDaysUntil = 0 Then
                lblMessage.Text = "Happy Halloween!"
            ElseIf iDaysUntil = 1 Then
                lblMessage.Text = "Tomorrow is Halloween!"
            Else
                lblMessage.Text = "There are only " & iDaysUntil _
                    & " days left until Halloween!"
            End If
        End If
    End Sub

    Private Function DaysUntilHalloween() As Integer
        Dim dtmHalloween As Date = New DateTime(DateTime.Today.Year, 10, 31)
        If DateTime.Today > dtmHalloween Then
            dtmHalloween.AddYears(1)
        End If
        Dim tsTimeUntil As TimeSpan = dtmHalloween - DateTime.Today
        Return tsTimeUntil.Days
    End Function

End Class
```

Description

- A content page can access a control in the master page if you expose the control as a public property in the master page. To do that, you code a property procedure with get and set procedures.

Figure 3-8   How to expose a master page control as a public property

How to access a public property of the master page from a content page

Figure 3-9 shows how you can access a public property in a master page from a content page. As you can see in the top part of this figure, you use the MasterType directive in the aspx file of the content page to specify the name of the type used for the master page. The value you name in this directive specifies the type of the object returned by the content page's Master property. So in this example, the Master property will return an object of type MasterPage. If you look at the class declaration in the previous figure, you'll see that MasterPage is the name of the class that defines the master page.

The second part of this figure shows two procedures from the code-behind file for the Order.aspx content page. As you can see, the Page_Load procedure calls a procedure named DisplayCartMessage. This procedure determines the number of items currently in the shopping cart and sets the Text property of the label exposed by the master page's MessageLabel property accordingly. But note that no value is assigned to the message label if the shopping cart is empty. In that case, the Page_Load procedure for the master page will set the label to the number of days remaining until Halloween.

Although using the MasterType directive in the content page's aspx file makes it easier to access the properties of the master page, you should realize that this directive isn't necessary. If you don't specify the MasterType directive, the Master property will return an object of type Master. You can then cast this object to the actual type of your master page to access any properties you've created.

For example, you could use code like this to assign text to the MessageLabel property:

```
If Cart.Count = 1 Then
    CType(Me.Master, MasterPage).MessageLabel.Text _
        = "There is one item in your cart."
ElseIf Cart.Count > 1 Then
    CType(Me.Master, MasterPage).MessageLabel.Text _
        = "There are " & Cart.Count & " items in your cart."
End If
```

Here, the Master object is cast to MasterPage so its MessageLabel property can be accessed. The purpose of the MasterType directive is to avoid this awkward casting.

A portion of the Order.aspx page

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Order.aspx.vb"
Inherits="Order" %>

<%@ MasterType TypeName="MasterPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="Main" Runat="server">
.
.
.
</asp:Content>
```

Two procedures from the code-behind file for the Order.aspx page

```
Protected Sub Page_Load(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
    If Not IsPostBack Then
        ddlProducts.DataBind()
        Me.DisplayCartMessage()
    End If
    SelectedProduct = GetSelectedProduct()
    lblName.Text = SelectedProduct.Name
    lblShortDescription.Text = SelectedProduct.ShortDescription
    lblLongDescription.Text = SelectedProduct.LongDescription
    lblUnitPrice.Text = FormatCurrency(SelectedProduct.UnitPrice)
    imgProduct.ImageUrl = "Images\Products\" _
        & SelectedProduct.ImageFile
End Sub

Private Sub DisplayCartMessage()
    Dim Cart As SortedList = CType(Session("cart"), SortedList)
    If Not Cart Is Nothing Then
        If Cart.Count = 1 Then
            Me.Master.MessageLabel.Text _
                = "There is one item in your cart."
        ElseIf Cart.Count > 1 Then
            Me.Master.MessageLabel.Text _
                = "There are " & Cart.Count & " items in your cart."
        End If
    End If
End Sub
```

Description

- The MasterType directive in an aspx file specifies the name of the master page type. If you include this directive in a content page, you can use the Master property in the code-behind file to access the exposed property of the master page.

Figure 3-9   How to access a public property of the master page from a content page

How to use nested master pages

In some web applications, particularly large ones, one or more sections of the web site may have common formatting elements in addition to the elements that apply to the entire site. In that case, you can use nested master pages as described in the following topics.

How nested master pages work

Figure 3-10 shows how *nested master pages* might be used in the Halloween Store application. Here, a portion of the web site is devoted to presenting do-it-yourself project instructions that show the user how to create home-made Halloween decorations and props. In addition to the other elements from the master page, each of the pages in this section of the web site has an additional banner that displays the text "Do-It-Yourself Projects."

To create this additional banner, a second master page named ProjectsMaster.master is used. The content page (in this example, Tombstone.aspx) specifies that its master page is ProjectsMaster.master. Like any other master page, the ProjectsMaster master page includes a content placeholder. However, unlike regular master pages, the ProjectsMaster master page also includes a Master directive that specifies the application's main master page, MasterPage.master. Thus, ProjectsMaster.master is nested within MasterPage.master. As you can see, elements from all three pages— MasterPage.master, ProjectsMaster.master, and Tombstone.aspx—are combined to create the final page that's sent to the browser.

A master page that's nested within another master page is called a *child master*, and the master page it's nested in is called a *parent master*. Note that a parent master can also be a child master. In other words, you can nest master pages more than one level deep. However, few applications require more than one level of nesting.
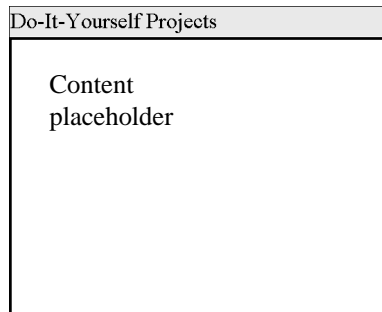
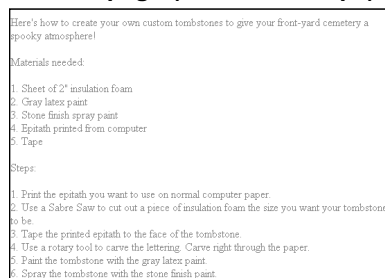## An application with nested master pages

Parent master page (MasterPage.master)
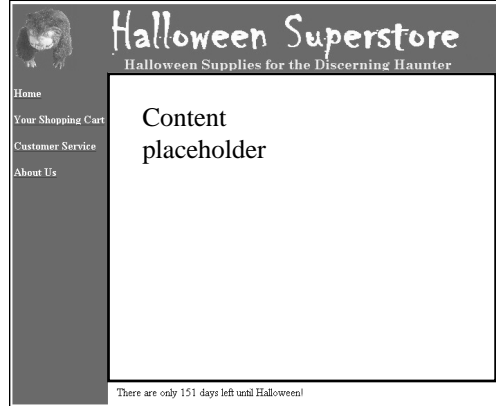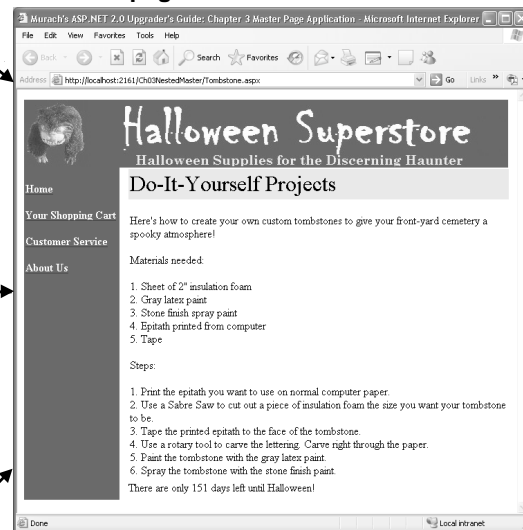
Child master page (ProjectMaster.master)

Rendered page

Content page (Tombstone.aspx)

Description

- Master pages can be nested. This lets you create elements that are common to all pages of a web site and other elements that are common to a subset of pages within the site.

- When you nest master pages, the content placeholder of one master page, called the *parent master*, holds another master page, called the *child master*.

Figure 3-10   How nested master pages work

How to create nested master pages

Unfortunately, Visual Studio 2005 doesn't support nested master pages in Design view. Microsoft has indicated that nested master pages may be supported in design view in a future version of Visual Studio. But until then, you'll have to work in Source view to create nested master pages and the content pages that use them. This is described in figure 3-11.

Note that no special coding is required to create a parent master page. As a result, the MasterPage.master page that was shown earlier in this chapter will work fine as a parent master page.

To create a child master page, first use the Add New Item command to add a new item to the web site and choose Master Page as the template for the new item. Then, in Source view, delete all the generated code except for the Master directive and the ContentPlaceHolder element. Change the ID attribute of the ContentPlaceHolder element as appropriate. In this figure, the ID of this element is set to Project.

Next, add a MasterPageFile attribute to the Master page directive. The value of the MasterPageFile attribute should be the name of the page you want to use as the parent master page.

Finally, add a Content element to the master page so that it contains the ContentPlaceHolder element. The ContentPlaceHolderID attribute of the Content element should name the ContentPlaceHolder element in the parent master page. In this example, the ContentPlaceHolderID attribute specifies Main as the name of the content placeholder. If you look back to figure 3-4, you'll see that Main is the name of the content placeholder in MasterPage.master.

Once you've created a child master page, you can create a content page as shown in the second code example in this figure. Here, the MasterPageFile attribute for the Page directive specifies the name of the child master page (ProjectsMaster.master). In addition, the ContentPlaceHolderID attribute of the Content element specifies Project, which matches the ID attribute of the ContentPlaceHolder element in ProjectsMaster.master.

Frankly, the lack of Design view support for nested master pages is a major impediment to their use. So if your application requires nested master pages, be prepared to do most of your development work without the benefit of Design view.

A child master page (ProjectsMaster.aspx)

```
<%@ Master Language="VB" CodeFile="ProjectsMaster.master.vb"
Inherits="ProjectsMaster" MasterPageFile="~/MasterPage.master"%>

<asp:Content runat=server ContentPlaceHolderID=Main>
    <table border=0 cellpadding=2 cellspacing=0>
        <tr>
            <td height=40 bgcolor=gainsboro>
                <span style="font-size: 24pt">Do-It-Yourself Projects</span>
            </td>
        </tr>
        <tr>
            <td valign=top>
                <asp:contentplaceholder id="Project" runat="server">
                </asp:contentplaceholder>
            </td>
        </tr>
    </table>
</asp:Content>
```

A content page that uses a child master (Tombstone.aspx)

```
<%@ Page Language="VB" MasterPageFile="~/ProjectsMaster.master"
AutoEventWireup="false" CodeFile="Tombstone.aspx.vb" Inherits="Tombstone" %>

<asp:Content ID="Content1" ContentPlaceHolderID="Project" Runat="Server"><br />
    Here's how to create your own custom tombstones to give your front-yard
cemetery a spooky atmosphere!<br /><br />
    Materials needed:<br /><br />
    1. Sheet of 2" insulation foam<br />
    2. Gray latex paint<br />
    3. Stone finish spray paint<br />
    4. Epitath printed from computer<br />
    5. Tape<br /><br />
    Steps:<br /><br />
    1. Print the epitath you want to use on normal computer paper.<br />
    2. Use a Sabre Saw to cut out a piece of insulation foam the size you want
your tombstone to be.<br />
    3. Tape the printed epitath to the face of the tombstone.<br />
    4. Use a rotary tool to carve the lettering. Carve right through the
paper.<br />
    5. Paint the tombstone with the gray latex paint.<br />
    6. Spray the tombstone with the stone finish paint.
</asp:Content>
```

Description

- Because the Web Forms Designer doesn't support nested master pages, you have to work in Source view to create child masters and content pages that use a child master.

- To create a child master page, add a MasterPageFile attribute to the Master page directive of a master page. Then, delete the rest of the generated code except for the ContentPlaceHolder element, and create a Content element that includes the ContentPlaceHolder element.

- To create a content page that uses a child master, specify the child master in the content page's MasterPageFile element. Then, add a Content element with the content for the page, and set the ContentPlaceHolderID attribute to the ID of the child master's content placeholder.

Figure 3-11   How to create nested master pages

Perspective

I hope this chapter has illustrated the power of master pages. In fact, I recommend that you use master pages for all but the simplest applications, even if you start out with nothing in your master pages but placeholders. Then, when you're ready to provide a professional look to your content pages, you can enhance the master pages, which will also enhance all of your content pages.

The alternative is to convert regular content pages so they use the master pages that you develop later on. But as figure 3-6 shows, that's a time-consuming and error-prone procedure. How much better it is to think ahead.

New terms

master page
content page
content placeholder
nested master pages
parent master
child master