

Beginning SQL Server 2005 Express Database Applications

with Visual Basic Express and
Visual Web Developer Express
From Novice to Professional



Rick Dobson

**Beginning SQL Server 2005 Express Database Applications
with Visual Basic Express and Visual Web Developer Express From Novice to Professional
Copyright © 2006 by Rick Dobson**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN (pbk): 1-59059-523-8

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editors: Tony Davis and Matthew Moodie

Technical Reviewer: Cristian Lefter

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser

Project Manager: Beth Christmas

Copy Edit Manager: Nicole LeClerc

Copy Editors: Damon Larson and Freelance Editorial Services

Assistant Production Director: Kari Brooks-Copony

Production Editor: Kelly Winkist

Compositors: Dina Quan and Diana Van Winkle, Van Winkle Design Group

Proofreader: April Eddy

Indexer: Valerie Perry

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com> in the Source Code section. You will need to answer questions pertaining to this book in order to successfully download the code.

User Controls,

Master Pages,

and Login Controls

“Easy yet powerful” is a goal often claimed for computer innovations but very rarely achieved. The improvements in ASP.NET 1.0 relative to ASP and ASP.NET 2.0 versus ASP.NET 1.0 are both substantial, but the enhancements from ASP to ASP.NET 2.0 are truly monumental. This chapter drills down on three innovations introduced with ASP.NET 2.0 that enable beginning and mid-level developers to manage Web sites with both power and ease never before possible. Even decision analysts will be able to take advantage of these new capabilities as they make the results of their efforts available to their clients – the senior decision-makers in organizations.

The first innovation covered in this chapter is a new implementation of user controls, which serve as containers for custom controls. Custom controls bring value to solutions by being more highly tuned for a specialized task than a built-in control or by coordinating several built-in controls to perform a task that none of the built-in controls can achieve individually. ASP.NET 2.0 user controls radically simplify the creation and re-usability of custom controls. This chapter’s coverage of user controls reviews the creation and re-use of three custom controls to familiarize you with different approaches for taking advantage of user controls.

The second innovation, master pages, targets developing a consistent look across all, or a subset of, the pages at a site. Master pages can be a rich topic, but even a very shallow understanding of this topic can generate a big payoff in the way your site looks to visitors. This chapter trains you in how to create and use master pages, with a special focus on how to use the Menu control on a master page to simplify Web site navigation.

Web site security used to be a reasonably advanced topic with ASP and even ASP.NET 1.0. However, the new Login controls with ASP.NET 2.0 can reduce Web site security to adding one or two controls to a Web page. The Login controls work in coordination with the ASP.NET Web Site Administration Tool. This browser-based, graphical tool dramatically lowers the complexity of creating new users and managing existing users, enforcing role-based security, and specifying access rules for folders at a Web site. This chapter’s coverage of Login controls will give you the background you need to implement Web site security in a way that’s easy to learn and apply.

User Controls

Visual Web Developer Express (VWDE) has its own impressive array of built-in controls for Web forms. However, if you do much development with Web forms, it is likely that you will encounter a situation where a new, custom form control can help the clients of your application or improve your own productivity. If you ever hesitated when thinking of using custom controls in the earlier version of ASP.NET, now is the time to change your mind and start learning about this easy-to-use and productive feature.

While you can build a form from scratch with custom graphics for a control, it is more likely that you will start out with user controls by customizing the built-in Web form controls. Useful customizations can be as simple as pre-setting one or two properties for a built-in control. This kind of user control makes sense when you find yourself regularly setting some properties whenever you use the built-in Web form controls.

Another advantage of a user control is that it lets you standardize how to adapt the built-in controls. If you or your team regularly customizes controls, it is easy to end up with several different kinds of customizations throughout a project. This can even happen with a single developer who returns to work on a project after not working with the project for an extended period of time. Inconsistency in how a Web application customizes built-in Web form controls can make an application look unprofessional or even make an application confusing for users. By creating and sharing a common set of user controls, all developers can start out with the same set of custom controls.

Note User controls are not optimized for use across multiple users and projects. However, it is easy to copy-and-paste user controls between projects. I noticed several articles describing more rigorous approaches, such as creating an assembly with user controls and then referencing the assembly in one or more Web site projects or putting the assembly in the Global Assembly Cache on a computer. The benefits of these more advanced solutions can be offset by the difficulty of implementing them and living within their constraints. Try searching your favorite search engine for sharing user controls to discover documentation of advanced approaches for “sharing user controls”.

If you like programming with Visual Basic .NET, but you are not really a programming wizard, you can still pack a lot of custom behavior into user controls. With custom programming, you can extend the feature set of the built-in controls as well as enable multiple controls to work together. The programming works similarly to code behind a form, but your program instead resides in a module behind a user control.

This section performs two main functions. First, it shows how to create and use the most important user control features. This coverage equips you to go on and build your own custom controls. Second, it describes three user controls. You’ll learn how to use these in your custom solutions. The description and demonstration of each control will teach you about user controls.

- * The first control introduces the process for creating and invoking user controls with a sample that shows how to create a TextBox control for passwords. This custom control merely changes one property of a TextBox control for use on a Web form.
- * Next, the section embellishes the initial user control by adding code to leverage the custom property setting for a built-in TextBox control in the first user control sample.
- * The coverage of user controls wraps up with a sample that merges a DropDownList control and a Calendar control into a single user control:
 - * This last user control lets site visitors look up the date for a holiday by its name.
 - * The user control also computes the number of days to a holiday.

- * You can readily adapt this kind of sample for helping your site visitors learn about the important events associated with your business, such as the date to the next seminar or the date to the next version of your product and the rollout steps along the way.

Creating and Using a TextBox for Passwords

All the samples for this chapter were created in the WebSiteChapter11 Web site. This is a Web site built on the localhost. You can see the New Web Site dialog box for creating the chapter's Web site in Figure 11-1.

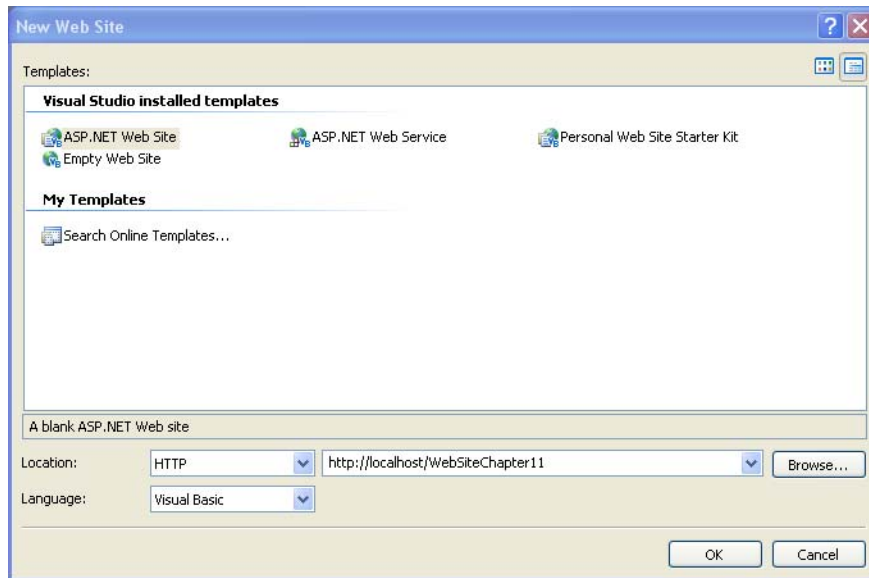


Figure 11-1. This New Web Site dialog box shows the settings used to create the Web site for the samples in this chapter.

A user control is actually a class that you can create using the ASP.NET user interface. Adding custom code to a user control can definitely improve its flexibility, but you do not need to add custom code to have a highly beneficial user control. The sample for this section changes one of the property settings of a TextBox control. You can use a TextBox control to collect a password from a user. While TextBox controls should generally show what a user types, a TextBox control for collecting a password can keep a password private by not showing the characters that a user types.

The `TextMode` property for a TextBox control determines how the control displays text that a user enters into it. There are three options.

- * The `SingleLine` setting allows the TextBox to show all user input on a single line. All characters typed while the TextBox has focus appear in the TextBox control. This is the default setting for the `TextMode` property.

- * The **Password** setting for the **TextMode** property makes the control suitable for collecting passwords. Input values do not show the characters typed—a substitute dot character shows that a character was typed. Selecting a **Password** setting from the Properties window for a **TextBox** automatically shortens the **Width** property relative to a **TextBox** with a **SingleLine** setting.
- * The third **TextMode** setting is **MultiLine**. This mode of input performs word wrap within a **TextBox** control and permits horizontal and vertical scrolling. This setting is not of interest for this sample because a **Password** setting enables single-line input.

Creating the Password TextBox User Control

You can start to create a simple user control by adding a Web user control file to your Web site. The steps for doing this appear below. Figure 11-2 displays an Add New Item dialog box that shows the result of following the instructions.

- * Right-click the project in Solution Explorer and choose Add New Item.
- * In the Add New Item dialog box, select the Web User Control template.
- * Assign a name for the user control, such as **PasswordTextBox**. VWDE automatically assigns an **.ascx** extension to the name that you specify for the file. You can see the extension in Solution Explorer. The **.ascx** extension is a special one to denote files containing a user control.
- * If you are going to add code, you can select the Place code in a separate file box, but this selection is optional even if you do customize a control with the aid of code.

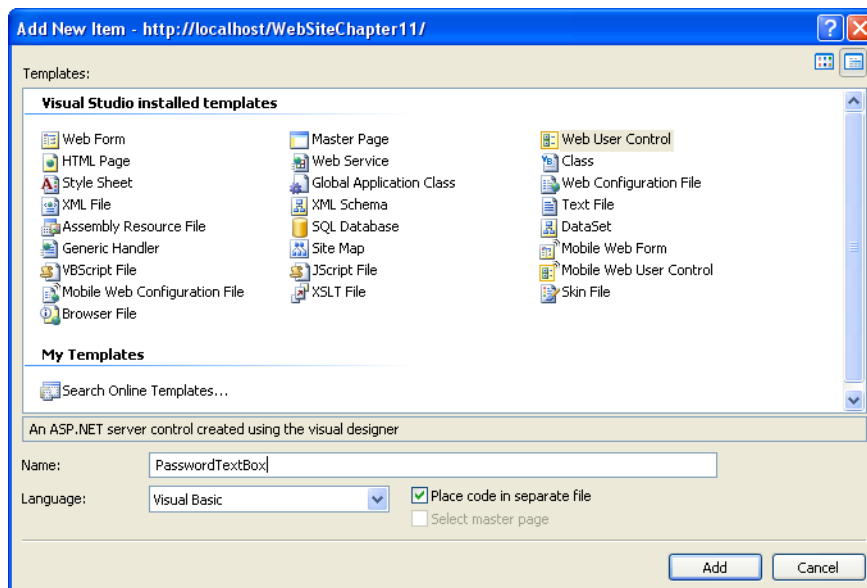


Figure 11-2. Add a new item to a Web application based on the Web User Control template for a new user control.

The .ascx file has a user interface that is similar to the one for a Web form. You can drag controls on to the file's Design or Source tabs. In addition, you can select controls, and assign property settings to them in the Properties windows. If you right click a blank area in the Design or Source tabs and choose View Code, you can open a code module behind the Design and Source tabs.

In addition to the similarities between .aspx and .ascx files, there are also some differences. For example, no default HTML code appears on the Source tab. The Source tab starts with a @Control directive instead of a @Page directive. The @Control directive has several attribute settings that assist in specifying the control. The following summary covers the attributes for the @Control directive for the PasswordTextBox control.

- * The Language attribute equals VB because Visual Basic is the default language for the control in Figure 11-2.
- * The AutoEventWireup attribute is false because of the selection of the Place code in a separate file box in Figure 11-2. If you fail to select a code-behind-control model, the AutoEventWireup attribute setting is true.
- * The CodeFile attribute equals the file name for control followed by .vb – namely, PasswordTextBox.ascx.vb.
- * The Inherits attribute equals PasswordTextBox. This is the name of the class for the PasswordTextBox control. The code module behind the Design and Source tabs contains a partial class declaration with the same name (PasswordTextBox). The contents of the PasswordTextBox.ascx and PasswordTextBox.ascx.vb files comprise the user control class.

You can complete the creation of the PasswordTextBox user control with a few more steps. Drag a TextBox from the Toolbox to the Source tab below the @Control directive. By default, the ID property of this TextBox control is TextBox1. From the Design tab, select the TextBox1 object in the top drop-down box within the Properties window. Figure 11-3 shows the TextMode property setting for TextBox1 in the PasswordTextBox user control in the Source tab and Properties window. A changed setting in either location propagates to the other location. After you make your property settings for the user control and add any code you need, don't forget to save the .ascx file to commit your changes.

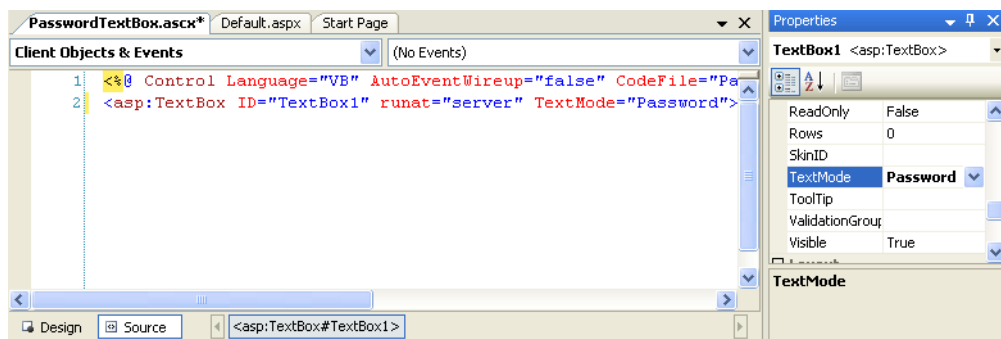


Figure 11-3. You can specify a user control as simply as dragging a control from the Toolbox to the Source tab and making just one Property setting.

Using the Password TextBox User Control

You cannot run a user control directly. Instead, you must add a user control to a Web form page on its Design tab. Recall that you can drag the built-in controls from the Toolbox to either the Design or Source tabs. Rather than drag a user control from the Toolbox to the Design tab for a Web form page, you drag a user control from Solution Explorer.

You can test the `PasswordTextBox` user control by adding a new Web form page to a ASP.NET project, such as `PasswordTextBoxDemo1.aspx`. Figure 11-4 shows the `PasswordTextBoxDemo1` page in the process of construction. In fact, all the controls are on the page, except for the `PasswordTextBox` control. The screen shot in Figure 11-4 shows the Design tab after the `PasswordTextBox` icon is dragged unto the page between some text (Enter password:) and a Button control. Releasing the mouse button adds the user control to the page with its smart tag box open. Click the control's left facing arrow to close the box.

The layout in Figure 11-4 facilitates a comparison of a built-in `TextBox` control and the `PasswordTextBox` user control. The text before each of the text boxes on the page says to enter a password. However, the top `TextBox` control on the form uses the default `TextMode` property setting of `SingleLine`. The `PasswordTextBox` user control has its `TextMode` property set to `Password` by default.

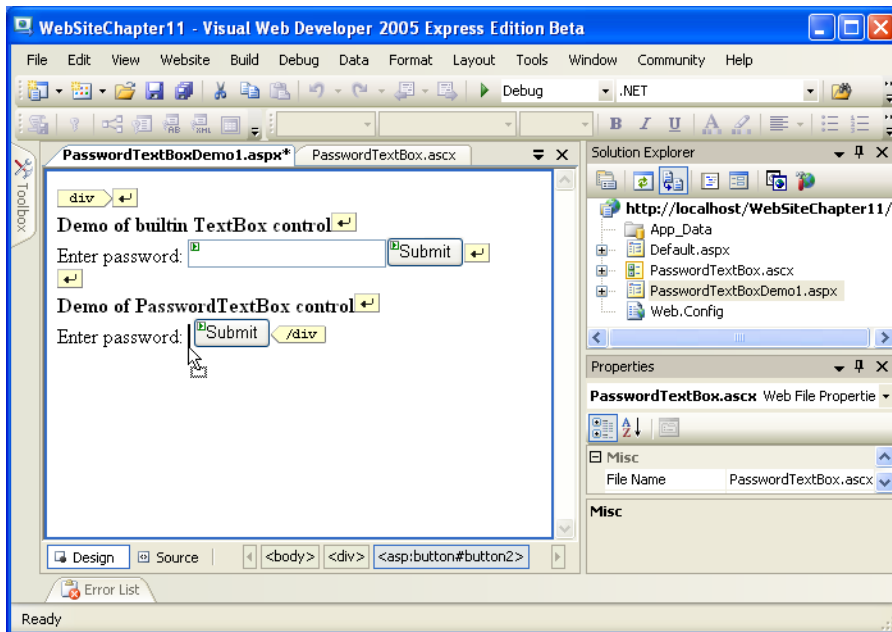
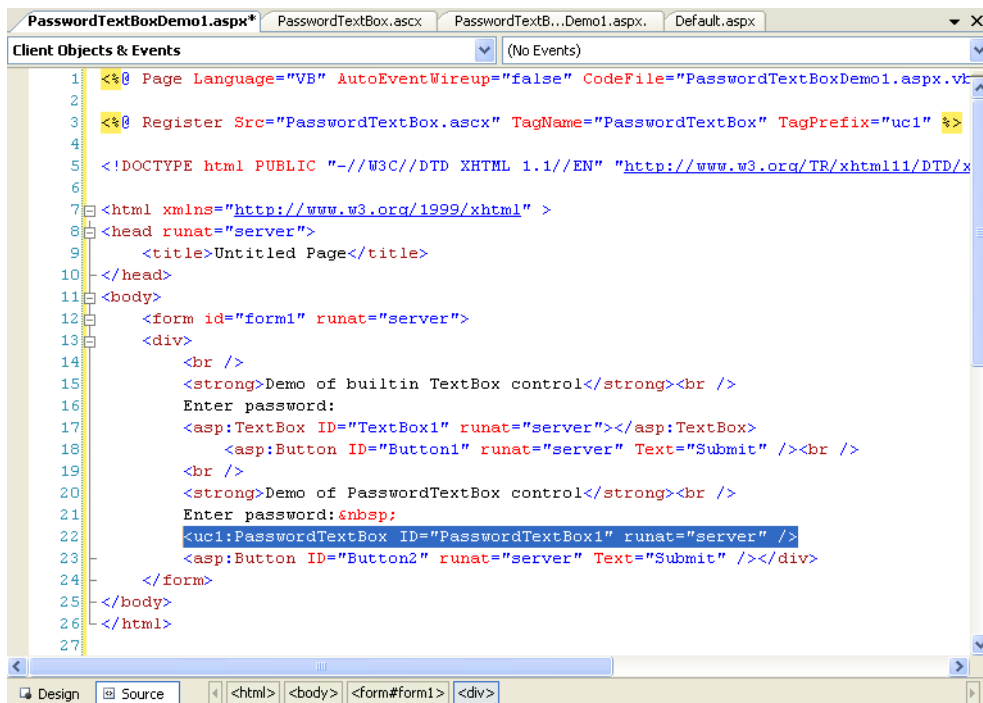


Figure 11-4. You can add a user control to a Web form page by dragging the control from Solution Explorer to the page's Design tab.

Figure 11-5 shows an excerpt from the Source tab for the `PasswordTextBoxDemo1` page. Notice the `@Register` directive. This directive provides the information qualifying the `PasswordTextBox` user control for use on the page. The directive has three attributes. The first attribute (`Src`) designates the file for the user control (`TextBoxPassword.ascx`). The second and third attributes (`TagName` and `TagPrefix`) specify the two-part name for the user control. Recall that built-in ASP.NET controls have a two-part name of `asp:controlname`, such as `asp:Button`. User control instances on a page have names composed of the values for the `TagPrefix` and `TagName` attribute values with a colon delimiting the two values.

The tag for the `PasswordTextBox` user control is highlighted in Figure 11-5. All instances of a user control on a page will have the same `TagPrefix` and `TagName` parts. Multiple instances must differ by their `ID` attribute value, which is the `ID` property setting in the Properties window. The `ID` attribute value for the user control in Figure 11-5 is `PasswordTextBox1`. You can override this default setting from either the Source tab or the Properties window. If you add a second `PasswordTextBox` user control to the `PasswordTextBoxDemo1` page without changing the name for the first user control, `VWDE` assigns a default `ID` attribute value of `PasswordTextBox2` for the second user control.

Note Recall that the `ID` attribute for a control corresponds to its name in a Visual Basic .NET procedure.



```
1 <%@ Page Language="VB" AutoEventWireup="false" CodeFile="PasswordTextBoxDemo1.aspx.vb"
2
3 <%@ Register Src="PasswordTextBox.ascx" TagName="PasswordTextBox" TagPrefix="uc1" %>
4
5 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/x
6
7 <html xmlns="http://www.w3.org/1999/xhtml" >
8 <head runat="server">
9 <title>Untitled Page</title>
10 </head>
11 <body>
12 <form id="form1" runat="server">
13 <div>
14 <br />
15 <strong>Demo of builtin TextBox control</strong><br />
16 Enter password:
17 <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
18 <asp:Button ID="Button1" runat="server" Text="Submit" /><br />
19 <br />
20 <strong>Demo of PasswordTextBox control</strong><br />
21 Enter password:&nbsp;
22 <uc1:PasswordTextBox ID="PasswordTextBox1" runat="server" />
23 <asp:Button ID="Button2" runat="server" Text="Submit" /></div>
24 </form>
25 </body>
26 </html>
27
```

Figure 11-5. After dragging a user control to a Web form page, `VWDE` adds a `@Register` directive to the page that references the control.

Figure 11-6 shows the `PasswordTextBoxDemo1` page open in a browser. You can open the page this way by choosing `Debug ~TRA Start Without Debugging` or by right-clicking the page in `Solution Explorer` and choosing `View in Browser`. Both the built-in `TextBox` control on top and the `PasswordTextBox` user control below it have the letters `abc` typed into them. Notice that the `PasswordTextBox` user control hides the value in the user control, but the built-in `TextBox` shows its entry.

The built-in `TextBox` control can be made to conceal its contents too by setting its `TextMode` property to `Password`. However, you must perform this extra step (set the property) to achieve this desirable outcome for a text box that is to collect a password. The `PasswordTextBox` user control requires no extra step to make it hide its contents. The more

properties you routinely set for selected built-in controls, the more valuable a user control is as a way of eliminating the need to set the properties and remembering to set one or two property settings for a control.

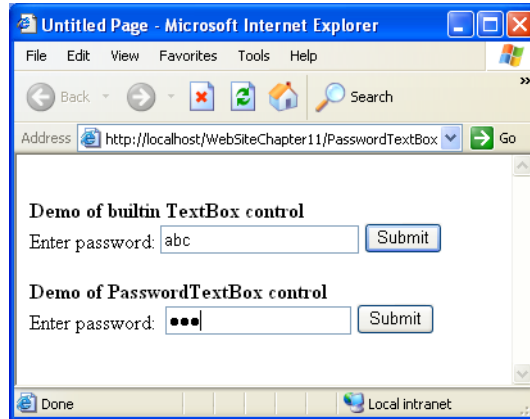


Figure 11-6. A PasswordTextBox control instance automatically hides values typed into it.

Adding Methods to a User Control

The PasswordTextBox user control pre-sets a non-default setting for a built-in control's property -- namely, the TextMode property of the built-in TextBox control. This type of capability is particularly useful whenever you want to format a control by controlling such features as its size, color, border -- in short, any property that you can set at design time from the Properties window.

In contrast to designating property values at design time, you may need to change or monitor a property value at run time. You can program a solution to perform actions at run time. There are two approaches to placing code behind a user control.

- * First, you can develop the code so that it runs inside the user control. In this scenario, a client application does not have direct access to the elements of a user control, but a user of a client application can manipulate the user control to generate some result. With the PasswordTextBox user control, a user can change the Text property of TextBox1 inside the control by entering a new value in the TextBox and clicking a Submit button. However, there is no direct programmatic means to modify the Text property for TextBox1 in the user control.
- * Second, you can develop code that exposes properties and methods within the module behind a user control to another application that has an instance of the user control. Invoke the Public access modifier to expose a property or a method implemented by a function procedure, a sub procedure, or a field from within a user control to a host application for the control.

The initial version of the PasswordTextBox user control from above has a couple of weaknesses that some code behind the control can remedy. First and foremost, there is no way for a host application to programmatically determine what value a user typed into the user control. Second, there is no way to assess whether the user entered a valid password.

This section amends the `PasswordTextBox` user control by adding some code behind the control to address both deficiencies.

Adding Code behind the PasswordTextBox User Control

The `TextBox` control inside the `PasswordTextBox` control has the name `TextBox1`. Within the control, you can determine what a user typed into the control with the `Text` property value of `TextBox1`. User controls do not automatically expose internal control properties to a host application, but you can use a function procedure to provide read-only access to the `Text` property value of `TextBox1` within the `PasswordTextBox` user control.

The following listing shows the `GetText` function procedure in the module behind the `PasswordTextBox` control. This function procedure exposes a method that returns the current value of the `Text` property of `TextBox1` within the control. The `Public` keyword in the function declaration exposes the procedure as a method.

```
Public Function GetText() As String
```

```
    Return Me.TextBox1.Text
```

```
End Function
```

A second function procedure with the name `IsValid` in the `PasswordTextBox` module defines another exposed method. The `IsValid` method returns a Boolean value to indicate if a password entered into the `PasswordTextBox` user control is valid. Again, the `Public` keyword makes the function procedure visible inside the host application for the user control.

The details of how the function procedure determines the validity of a password is arbitrary. You can replace the code within the function procedure with whatever your application requires. In the demonstration sample, there are two rules. First, a password's first character must begin with one of six characters – namely, d, g, p, r, t, and v. Second, the password's second character must be convertible to a numeric value of zero through 9. You almost surely will want to use different criteria for assessing the validity of a password, but the `IsValid` function procedure is interesting, in part, because it shows the structure of how to assess the validity of a password.

Despite the arbitrary nature of the two criteria for determining the validity of a password, you may find them interesting because of their string procedure and character-manipulation features.

- * The `chr1st` and `chr2nd` variables are two variables with a `Char` data type for holding the first and second password characters respectively.
- * The `startchar` array is a one-dimensional array defining the legitimate starting characters for a password. The second `Dim` statement, which is for the `startchar` array, populates the array with six character values beginning with d and ending with v.
- * Built-in `Mid` functions extract the first and second characters from the password value, and a `CChar` function converts the return value from a `String` data type to a `Char` data type.
- * A `For...Each` loop passes through legitimate beginning characters in the `startchar` array.

- * If the loop detects a legitimate beginning character with a numeric value in the second character, the function returns a value of `True`.
- * Otherwise, control passes to the first statement after the loop, which returns a value of `False`.

```
Public Function IsValid() As Boolean

    Dim chr1st As Char, chr2nd As Char
    Dim startchar() As Char =
        New Char(5) {"d"c, "g"c, "p"c, "r"c, "t"c, "v"c}

    chr1st = CChar(Mid(Me.TextBox1.Text, 1, 1))
    chr2nd = CChar(Mid(Me.TextBox1.Text, 2, 1))

    For Each chr As Char In startchar
        If chr = chr1st And IsNumeric(chr2nd) Then
            Return True
        End If
    Next

    Return False

End Function
```

Invoking User Control Methods from a Host Application

You can invoke any exposed user control methods with the same syntax that you use for a method or property of any built-in control. The general syntax is `usercontrolname.methodname`. For example, a host application for the updated `PasswordTextBox` user control can determine if the current `Text` property value of the control is valid with an expression such as this one: `PasswordTextBox1.IsValid`.

The following excerpt is from the `Button2_Click` procedure in the `PasswordTextBoxDemo2.aspx` page. This page is identical to the `PasswordTextBoxDemo1.aspx` page with the exception of the `Button2_Click` procedure. The procedure runs when a user clicks the button next to the `PasswordTextBox` user control on the form.

The procedure writes out one of two messages to the page with the `Write` method of the `My.Response` object (you can also use the `Text` property of a `Label` control to provide user feedback). If the `IsValid` method returns a value of `True`, then the procedure writes a value of `Password OK`. Otherwise, the procedure indicates the password is not valid and shows the input value. This feedback permits a user to determine if they entered the value that they intended. In a normal application, you would do something more than acknowledge a password is valid if the `IsValid` function returns a value of `True`. For example, you might transfer control to another page to offer a menu of choices or show some data.

Caution The method example for the `PasswordTextBox` user control illustrates the use of a simple method with a very basic user control. In a production application, you might use the `PasswordTextBox` user control with a `TextBox` control for accepting a user identification code. If the password is invalid on its own or in combination with the user identification code, your application can return a message to reenter a valid user ID

and password without giving any specific information that the password is invalid.

```
If Me.PasswordTextBox1.IsValid Then
    My.Response.Write("Password OK.")
Else
    My.Response.Write("Password not OK. " & _
        "You entered: " & Me.PasswordTextBox1.GetText)
End If
```

Note The Password control element within the Login controls has a much richer set of features for managing access to a Web site. Therefore, you should generally use Login controls if you need a password to grant access to a Web site and its folders. See the "Login Controls and Membership Features" section for in-depth coverage of Login controls. However, some developers may prefer to program their own password solutions for the experience or because they want a password for some other purpose besides tracking users within a Web site, such as permission to use a feature (those who know the password can gain access to the feature).

Developing a Calendar-based User Control

The PasswordTextBox user control is a very basic control that is representative of many controls that you are likely to create. Each of these types of user control can save you a little time and effort in your development efforts. The collection of all your user controls can generate substantial savings.

Another type of user control will deliver some unique functionality. While this type of user control may save development time as you re-use them in more and more solutions, they are primarily of interest because they provide special functionality that you cannot easily duplicate. As a result, this type of control will typically be more complicated than the type of user control demonstrated in the preceding sections.

The calendar-based user control sample that this section presents lets a user select a holiday name from a series of holiday names in a DropDownList control. When a user selects a holiday from a list of holidays, the user control displays the holiday's date in a Calendar control and the number of days to the holiday from today in a TextBox control. The user control has two additional TextBox controls that display the date today and the holiday's date.

The sample user control is called the DaysTo user control because this name describes its central feature. You can use this kind of control in any scenario that can benefit from a count down to the number of days to an event, such as a sale at your site, one or more seminars, or a prize award. Whether or not you decide to take advantage of the sample as a user control, you can derive value from the coverage of how to use the Calendar control this way as well as how to associate a list of event names with corresponding dates.

When building this kind of user control, it will often be easier to prototype your logic in a sample Web form page. If you were to start building a control directly in a user control file (.ascx), you would need a host application Web form page to test it. Developing with a single file is often easier and faster than developing with two files. After you get the control layout and code behind the page working, you can create an empty user control and copy your prototype controls and code from the prototype Web form page to the user control file.

As a last step, you can create a host Web form page and test the user control in the host application.

Prototyping the DaysTo User Control

Figure 11-7 shows the Design tab for the PrototypeDaysTo Web form page. As you can see, this Web form page uses a table to help position the controls properly next to one another. You can add a table to a Web form page with the Insert Table item of the Layout menu. The table has one row and two columns. A DropDownList control is in the first column, and a Calendar control is in the right column. Below the Calendar control are three TextBox controls for the today's date, the selected holiday's date, and the number of days to the holiday from today.

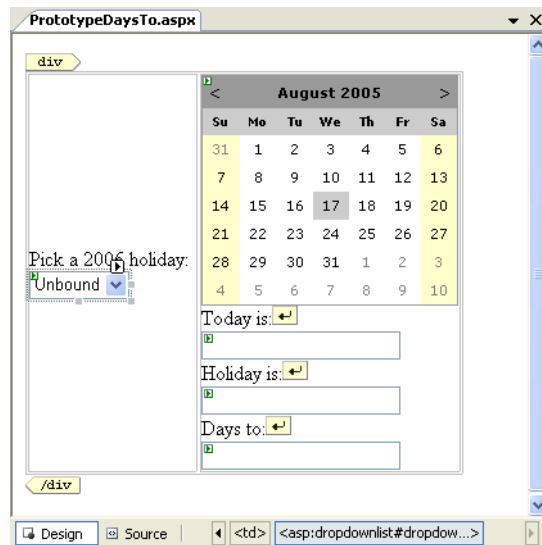


Figure 11-7. When building more complex user controls, it is often helpful to start with a Web form page.

A view of the PrototypeDaysTo Web form page can help you to understand the following description of the application. Figure 11-8 shows the Web form page after it initially opens. When the page first opens, it computes the number of days from today to New Year's Day in 2006. Users can select another holiday, such as Thanksgiving Day, from the DropDownList control to show the date for the selected holiday on the Calendar and to compute the number of days from today to the selected holiday.

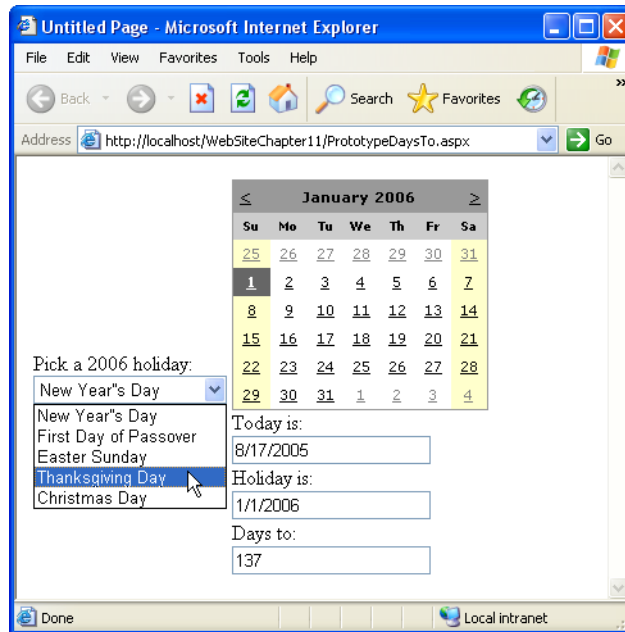


Figure 11-8. The *PrototypeDaysTo* Web form page lets a user pick a holiday and then updates the Calendar control view to show the holiday's date and the days to the holiday.

The code behind the Web form page in Figure 11-7 uses three procedures and two module-level declarations for a `DataTable` (`tblHolis`) and a `DataRow` (`drHoli`). The `tblHolis` `DataTable` holds a list of holidays from which users can pick. The three procedures respond to user selections, compute the days to a selected holiday, and manage the display of values in the Web form page.

Both `DataTable` and `DataRow` types belong to the `System.Data` namespace that's especially important for use with ADO.NET 2.0. In ADO.NET 2.0 solutions, you'll primarily use `DataTable` instances to store a collection of column values from a table or view in a database. A `DataRow` object often stores an individual row of column values from a `DataTable`. Chapter 12 demonstrates the use of ADO.NET 2.0 for use with SQL Server Express. The application in this section uses the `tblHolis` `DataTable` variable to store a local list of holidays. The `drHoli` `DataRow` variable assists with populating the `DataTable` as well as storing the date information for a holiday selected by a user.

You can think of a `DataTable` as an advanced kind of two-dimensional array. A `DataTable` is especially advantageous for this application because it lets you specify different data types for different columns within the table. Therefore, one column can have a `String` data type for storing holiday names, and three additional columns can have `Integer` data types to hold values for the day, month, and year of holidays.

The `Page_Load` event procedure is the first procedure to operate for the sample application. This procedure has two parts. First, the procedure specifies the data types for the `tblHolis` `DataTable` columns and populates the `tblHolis` `DataTable` with an initial set of holiday names and their dates in 2006. You can modify this list as your needs dictate. Second, the procedure populates a `DropDownList` control with a list of holiday names and picks an initial holiday to show when the Web form page displays the first time.

The following excerpt from the top of the `Page_Load` procedure shows the code for declaring the column data types in the `tblHolis` `DataTable`. When the `tblHolis` variable is

declared at the module level it is just declared as a `DataTable` type with no column specifications. The following block of code adds four columns to `tblHolis`: `Year`, `Month`, `Day`, and `Name`. The first three columns have an `Integer` data type. The last column has a `String` data type.

```
tblHolis.Columns.Add("Year", GetType(Integer))
tblHolis.Columns.Add("Month", GetType(Integer))
tblHolis.Columns.Add("Day", GetType(Integer))
tblHolis.Columns.Add("Name", GetType(String))
```

Immediately following the code to add columns to `tblHolis`, a series of code blocks add successive holidays to the `tblHolis` `DataTable`. The first of these code blocks, which adds New Year's Day in 2006, appears below. The code block starts by instantiating a new `DataRow` instance based on the columns of the `tblHolis` `DataTable`. Next, it populates the row with column values for the New Year's Day holiday. The code block concludes by adding the `drHoli` `DataRow` to the `Rows` collection of the `tblHolis` `DataTable`.

```
drHoli = tblHolis.NewRow()
drHoli("Year") = 2006
drHoli("Month") = 1
drHoli("Day") = 1
drHoli("Name") = "New Year's Day"
tblHolis.Rows.Add(drHoli)
```

Additional code blocks like the preceding one add other holidays to the `tblHolis` `DataTable`. The `Page_Load` procedure has code blocks for the following holidays: First Day of Passover, Easter Sunday, Thanksgiving Day, and Christmas Day. The code for an additional holiday (Mother's Day) is commented out. This commented out section will be used in a demonstration of the user control based on this prototype solution.

Note You can readily adapt this sample by replacing the holiday names and dates with whatever special days and dates you require. For example, if you wanted to compute the days to holidays in 2007, you could change the holiday dates to those for 2007.

The final code block in the `Page_Load` procedure populates the `DropDownList` control with holiday names and selects New Year's Day as a holiday to initialize the Web form page. You can see from the preceding code block that New Year's Day is the first row in the `tblHolis` `DataTable`. In order not to repeatedly populate the `DropDownList` control with new rows every time the page loads or to always make New Year's Day the selected holiday, this code occurs within an `If...Then` statement.

- * The condition for the `If` clause is that the page is not a post back from a browser session. If the page is not posting back from a browser session, then it is the first time the server sends the page to the browser, and the code therefore initializes the page.
- * The `SelectedIndex` property for a `DropDownList` control, such as `DropDownList1` on the Web form, denotes the currently selected item in the `DropDownList` control. Setting the property to zero selects the first row in the control.

- * The `ComputeDiff` procedure handles the update of the Calendar control and the TextBox controls for the currently selected holiday. This procedure is discussed more thoroughly subsequently.

```

If Me.IsPostBack = False Then

    For Each drHoli In tblHolis.Rows
        Me.DropDownList1.Items.Add(drHoli("Name"))
    Next
    Me.DropDownList1.SelectedIndex = 0
    ComputeDiff(CType(Me.DropDownList1.SelectedIndex))

End If

```

The `DropDownList1_SelectedIndexChanged` procedure contains a single line of code. This procedure fires whenever a user selects a new item from the `DropDownList` control. Passing the index value for the newly selected item updates the form for the holiday that a user selected most recently.

```

ComputeDiff(CType(Me.DropDownList1.SelectedIndex))

```

The `ComputeDiff` procedure manages the Web form's display based on a `Byte` parameter value passed to it. This parameter value points to a row in the `tblHolis` `DataTable`.

- * The procedure begins by passing the column values from the selected row in the `tblHolis` `DataTable` to the `drHoli` `DataRow`.
- * Next, a `DateSerial` function computes a date based on `drHoli` column values and assigns the date to the `SelectedDate` property of the calendar.
- * Then, the procedure changes the Calendar control's view to show the selected date.
- * The remaining code in the procedure populates the values in `TextBox1` through `TextBox3`.
 - * `TextBox1` shows the date for now or today.
 - * `TextBox2` shows the date for the selected holiday.
 - * `TextBox3` shows the days remaining to the holiday if the current date is not already past the holiday date. Otherwise, `TextBox2` and `TextBox3` inform the user that the holiday has passed.

```

Sub ComputeDiff(ByVal index As Byte)

    Dim ts1 As TimeSpan

    'Specify holiday to process and show it in calendar
    drHoli = tblHolis.Rows(index)
    Me.Calendar1.SelectedDate = DateSerial(CInt(drHoli("Year")), _
        CInt(drHoli("Month")), CInt(drHoli("Day")))
    Me.Calendar1.VisibleDate = Me.Calendar1.SelectedDate

    'Display results for selected holiday; start by
    'initializing date now

```

```

Me.TextBox1.Text = Now.ToString("d")
Me.TextBox2.Text = Me.Calendar1.SelectedDate.ToString("d")
ts1 = Me.Calendar1.SelectedDate - Today
If ts1.Days >= 0 Then
    Me.TextBox3.Text = ts1.Days.ToString
Else
    Me.TextBox2.Text = "holiday is gone"
    Me.TextBox3.Text = "pick another holiday"
End If

End Sub

```

Creating a Shell and a Host Application for the DaysTo User Control

After you've built and tested a prototype application for a user control, you are ready to perform two more steps. First, transfer the prototype application on a Web form page to a user control file (.ascx). Second, create a host page and drag the user control into the host page.

Before you can transfer the prototype for a user control, you need a new user control shell. Create a new shell by choosing Add New Item from the context menu for the Web site project in Solution Explorer. Choose the Web User Control template. Name the new user control shell DaysTo. This creates an empty DaysTo.ascx file.

There are two parts to copying the prototype controls on a Web form page to the user control shell.

- * First, you need to copy the control layout. You can do this by copying the layout of the controls from the Design tab in the prototype page to the Windows Clipboard. Next, you can copy the control layout to the Design tab in the user control file.
- * Second, you need to copy the code from the prototype page to the module within the DaysTo user control file. Open the module in the prototype page by right-clicking a blank area of the Design tab and choosing View Code. Then, copy the code from the prototype starting at the two module-level declarations and continuing through to the End Sub statement for the final procedure. Paste this code into the Partial Class DaysTo for the DaysTo module.

Now you are ready to create a host page. Start a new Web form page in the WebSiteChapter11 Web site and name the page HostDaysTo.aspx. From Solution Explorer, drag the icon for the DaysTo user control to the Design tab of the HostDaysTo Web form page. You are now ready to start testing how your host works with your user control.

Testing the DaysTo User Control and its Host Application

Figure 11-9 shows an excerpt from the Source tab of the HostDaysTo.aspx file. You can see the @Register directive for the DaysTo user control. Recall that VWDE automatically adds this directive after you drag a user control, such as the one for the DaysTo.ascx file, to a Web form page. Farther down the Source tab, you can see the DaysTo user control (uc1:DaysTo) within the div block.

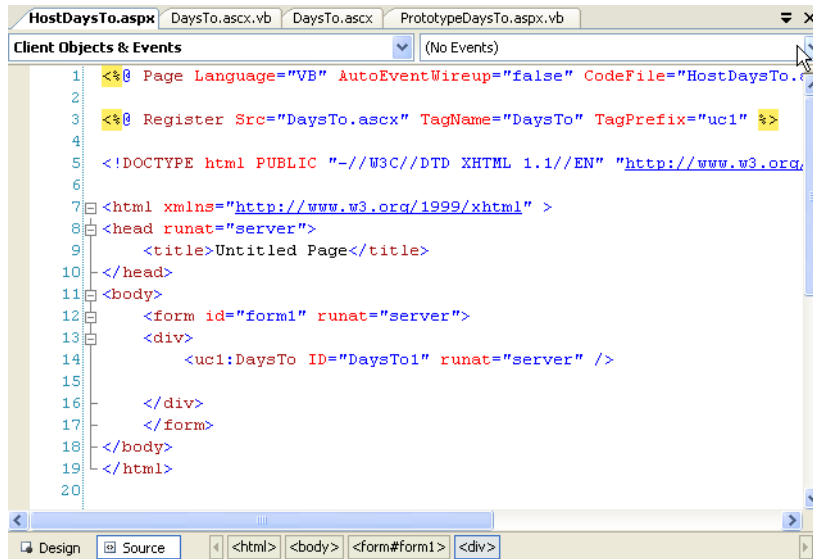


Figure 11-9. After you drag a user control to a Web form page, you are ready to test the user control on the Web form page.

Choosing Start Without Debugging from the Debug menu opens the HostDaysTo.aspx Web form page with the DaysTo user control. The form initially opens similarly to Figure 11-8 without the DropDownList control being open. After you select Thanksgiving Day from the DropDownList control, the Calendar control view changes to the one in Figure 11-10. Notice there are 463 days from the day the selection was made to Thanksgiving Day in 2006. Changing the selected holiday to Christmas Day 2006 shows the Calendar control advance by one month to show Christmas day. In addition, the code behind the DaysTo control re-computes the days to the holiday by adding 32 days for a total of 495 days to the holiday.

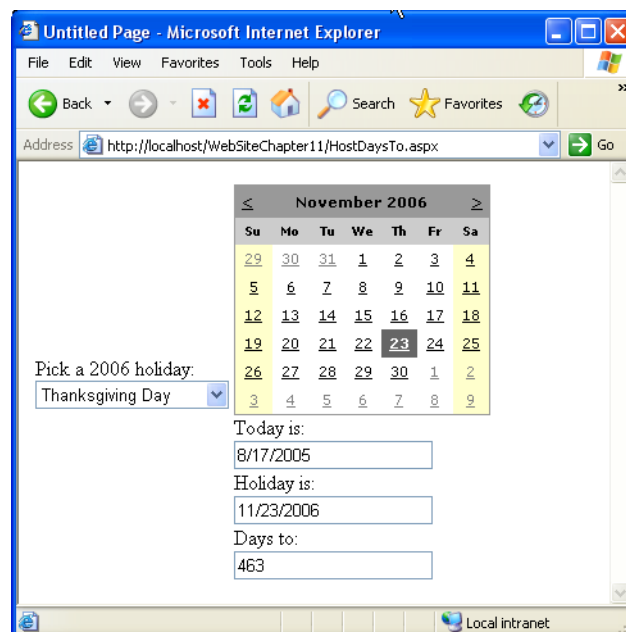
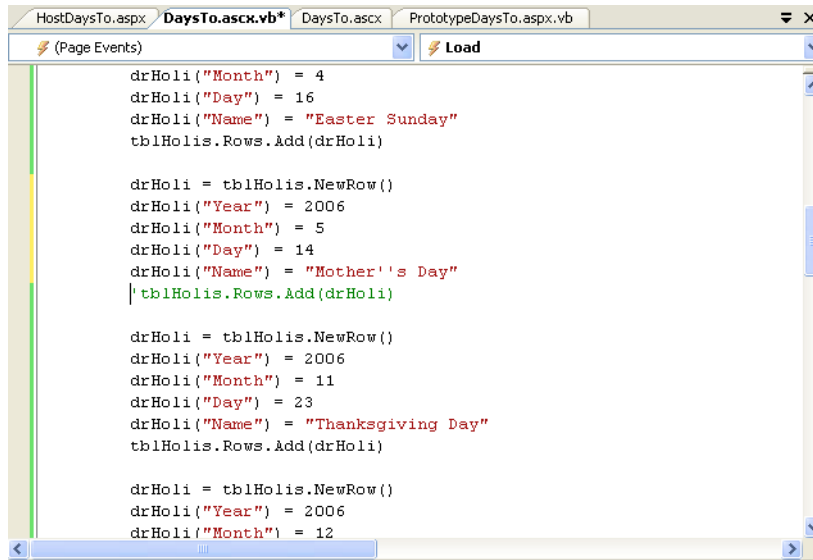


Figure 11-10. Choosing Thanksgiving Day as Figure 11-8 demonstrates alters the Calendar control view and the computed days to the holiday.

When you make a change in the user control module, the host web page reflects the revision the next time it opens. Figure 11-11 shows a change in process for the DaysTo user control code module. This modification involves removing comment markers for the Mother's Day holiday in 2006. After removing all comment markers for the new holiday, you can re-open the page from the host Web page. VWDE asks if you want to save your changes. Choosing Yes opens the page with your edits. In this case, the DropDownList control shows a new holiday for Mother's Day 2006.



```
HostDaysTo.aspx DaysTo.ascx.vb* DaysTo.ascx PrototypeDaysTo.aspx.vb
(Page Events) Load
drHoli("Month") = 4
drHoli("Day") = 16
drHoli("Name") = "Easter Sunday"
tblHolis.Rows.Add(drHoli)

drHoli = tblHolis.NewRow()
drHoli("Year") = 2006
drHoli("Month") = 5
drHoli("Day") = 14
drHoli("Name") = "Mother's Day"
|tblHolis.Rows.Add(drHoli)

drHoli = tblHolis.NewRow()
drHoli("Year") = 2006
drHoli("Month") = 11
drHoli("Day") = 23
drHoli("Name") = "Thanksgiving Day"
tblHolis.Rows.Add(drHoli)

drHoli = tblHolis.NewRow()
drHoli("Year") = 2006
drHoli("Month") = 12
```

Figure 11-11. You can make a change to a user control, and the modification takes effect the next time a host page for the user control opens.

Master Pages

Presenting a consistent look and feel across the pages of a Web site and enabling easy site navigation are two critical features for making your site attractive to visitors. A consistent look and feel for your pages helps to develop a brand identity for your Web site. Pages sharing a common look and feel reinforce a common identity. Easy navigation simplifies viewing additional pages at your site. One way of assessing brand loyalty is how many pages visitors view at a site. Therefore, making navigation easy also reinforces the brand for your Web site.

VWDE offers several approaches to developing a consistent look and feel along with easy site navigation. This section demonstrates the use of Master pages along with a Menu control to achieving these goals. Some of the prominent Master page features follow.

- * A Master page allows you to set a color scheme and format for displaying other pages at a site.
- * Web site visitors never view Master pages. Instead, they view pages based on a Master page.

- * You can add content to a Master page, such as a copyright note, that can appear on all the pages based on that Master page.
- * Placing a Menu control for site navigation on a Master page is one way to enable consistent navigation across all the pages based on a Master page.
- * By using two or more Master pages at a Web site, you can reinforce different brand identities for subsets of pages at a single site. This capability is especially valuable when a single Web site promotes multiple products or services.

This section introduces you to Master pages by presenting a walkthrough of several important Master page capabilities. You'll learn how to:

- * Create a Master page
- * Add graphics and text to a Master page
- * Create content pages that appear with the formatting for a Master page
- * Add a Menu control to facilitate navigation across the pages based on a Master page

Creating a Master Page

A Master page is a type of file at a Web site. The process for creating a Master page is similar to that for a Web form page, a user control, or an HTML page. You start to add a Master page to a Web site by right-clicking the Web site or root folder name in Solution Explorer and choosing Add New Item. In the Add New Item dialog box, select the Master Page template and designate a name for the file or accept the default name, such as **Master1.master**.

Figure 11-12 shows a view of the Source tab for a Master page named Master1 immediately after it is added to a file. Two distinctive features identify this type of a page.

- * First, there is a **@Master** directive instead of a **@Page** directive at the top. This directive specifies that the page inherits from the **Master1** class which is based on the **Master1.master** file and the **Master1.master.vb** files.
- * Second, the page contains a **ContentPlaceHolder** control. This control appears within the **div** block on the page. Other content pages based on this Master page will surround their content with the formatting surrounding the **ContentPlaceHolder** control on the Master page.

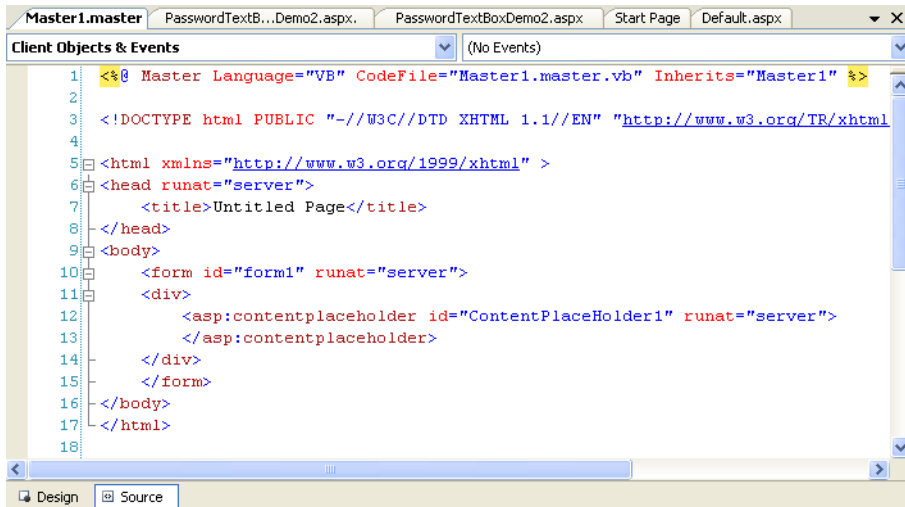


Figure 11-12. A Master page will always start with a `@Master` directive, and it should contain a `ContentPlaceHolder` control.

After you initially add a Master page to a Web site, you will need to format it before the page can add value to a Web application. One vital step is to add a table template that subdivides a Master page into parts, such as header, footer, and left border areas. You can format the various areas of a Master page with the Properties window or sometimes by dragging borders to resize areas. You will typically want to insert some standard text and graphics that appear on all content pages based on the Master page. This standard text and graphics will help to build a brand identity for the content pages based on the Master page.

Note You will eventually want to add a Menu control to most Master pages, but you do not need to perform this step until after you add content pages based on a Master page. You can't add content pages based on a Master page until you first create a draft of your Master page and save it.

You can add a layout table to a Master page by choosing Layout ~TRA Insert Table. In the Insert Table dialog box select the Template radio button. Then select a style of layout for your Master page. Figure 11-13 shows the Header, footer, and side template selected. Clicking OK adds the layout table to the page above the `ContentPlaceHolder` control.

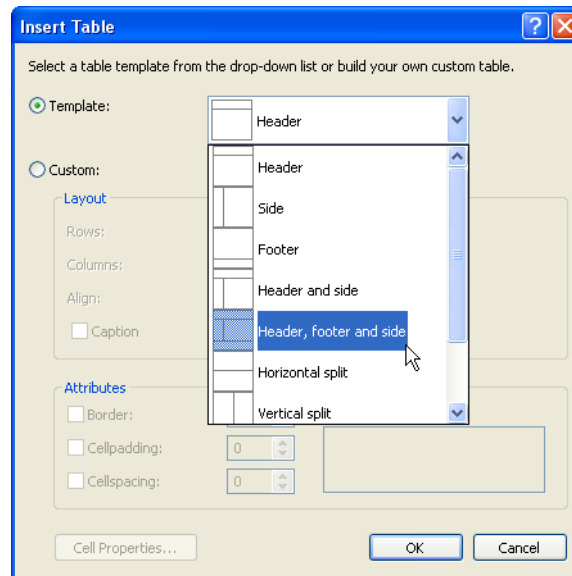


Figure 11-13. The Header, footer, side template is one of several layouts that you can designate for the design of a Master page.

Adding Graphics and Text to a Master Page

After you add a layout table to a page, you will want to format the table to develop a distinctive look for the content pages based on this Master page. The header, side, and footer areas act as cells within a table. You can select an area, and apply formatting, such as a background color, to the area through the Properties window. You can also add text and images to the different areas of a Master page. In addition, you can drag the ContentPlaceHolder control to an area within the layout table, such as the area bordered by the header, side, and footer areas.

I created a graphic image with the WordArt tool in Word. This tool allowed me to specify a shape, color, and other formatting for the letters in a stream of characters. The characters in the graphic image are: Visual Web Developer Express by. I aligned characters along a wave. Then, I processed the WordArt image further and saved the image as `VWDE.gif` with Paint. I also borrowed a picture of myself (`RD_Pix.jpg`) from my popular ProgrammingMSAccess.com site. Both the WordArt and picture image files were saved in the root folder of the WebSiteChapter11 Web site.

Figure 11-14 shows `Master1.master` in its Design tab after some preliminary formatting. The text in the footer area is partially cropped as are some property names from the Properties window. The selected table cell corresponds to the header area in the layout table.

- * The header area has a background color property (`BgColor`) set to black (`#000000`). A Color Picker dialog box offers several different ways for designating the background color.
- * The `VWDE.gif` and `RD_Pix.jpg` files were dragged into the header area from Solution Explorer. VWDE automatically wrapped the graphic images in Image controls.
- * The side area has a gray color setting.

- * The footer area has some formatted text within it. The text is set to have a font size of 14 points. After selecting the text, you can make this setting from the Formatting toolbar. There is no `BgColor` setting for the footer area.
- * The area bordered by the header, side, and footer areas has its vertical alignment (`VALign` property) set to `top`. Notice the `ContentPlaceHolder` control is at the top of the area. By default, the `VALign` property for any layout table area is `middle`. Therefore, the header, side, and footer areas have a middle vertical alignment property setting for the content within them.

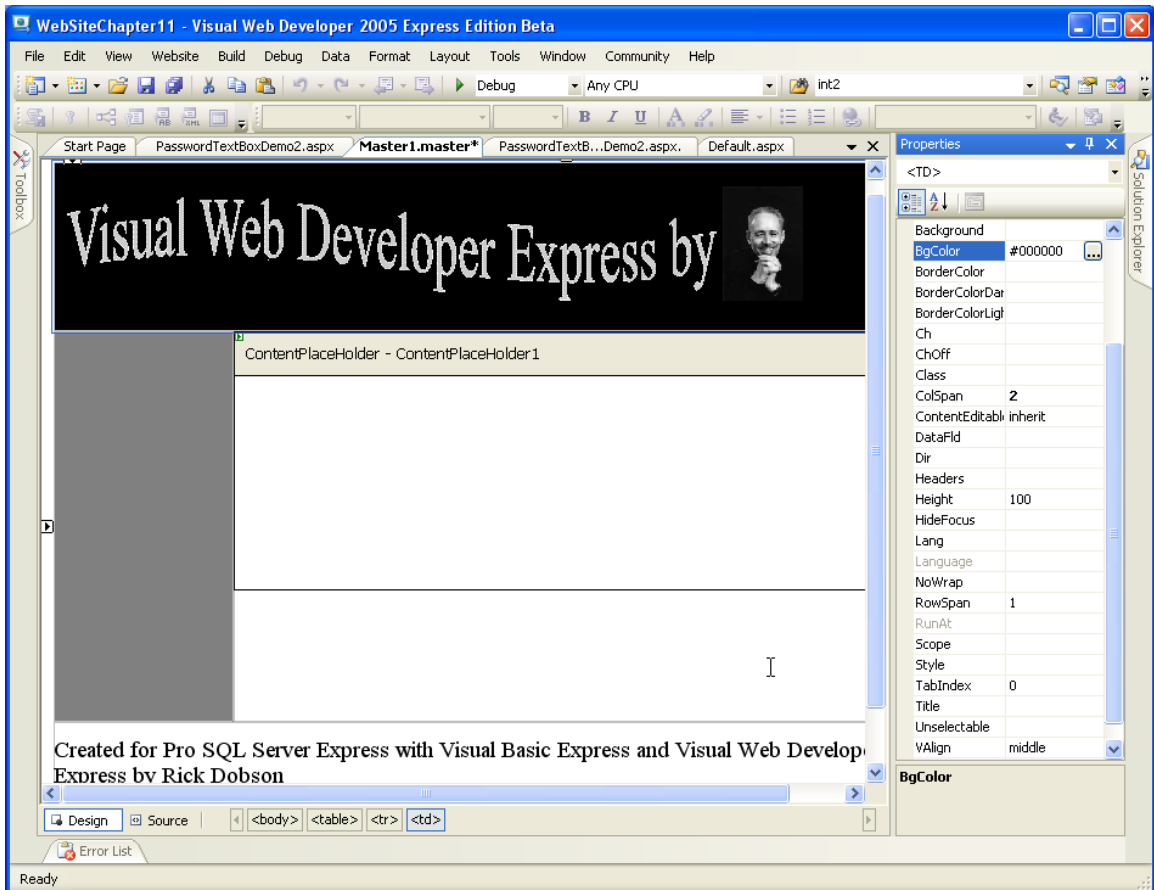


Figure 11-14. You will typically format a Master page by setting properties in the Properties window for layout table areas and by adding text and graphics to the areas.

Creating Content Pages

After you create a Master page, you can use it to create content pages based on it. This section describes the process for creating three content pages. Reviewing the process for creating these content pages will help you understand how to use content pages and extend your ability to create flexible hyperlinks, format text on Web pages generally, and develop solutions with user controls.

Two of the new content pages, `Home.aspx` and `About.aspx`, are typical of many Web applications. `Home.aspx` is a welcome page. It says hello to a site visitor and offers a link to

another site page. A Web application home page often offers many links to different pages throughout a site. The `About.aspx` page provides some general information about a site. It can offer a link for learning more or providing feedback. The `About.aspx` page in this section starts up the local email application on a computer with the `mailto` attribute of the HTML anchor tag.

Note The home page for an ASP.NET Web site will often have the name `default.aspx` in the Web site's root folder. We'll be using this page extensively in our coverage of Login controls later in this chapter. To keep the coverage of Master pages and Login controls independent, we designate `Home.aspx` as the home page for the Web site in this section. In actual practice, you are likely to want to use `default.aspx` instead of `home.aspx` as the home page for a site.

Both `Home.aspx` and `About.aspx` are new pages developed for use with the `Master1.master` page. You will probably encounter situations where you want to retrofit a previously created Web page for use with a subsequently created Master page. The `PasswordTextBoxDemo3.aspx` page is an adaptation of the previously discussed `PasswordTextBoxDemo2.aspx` page. This adaptation gives you experience with the issues associated with converting a page manually for use with a Master page. In particular, you'll gain experience with using a user control in a content page based on a Master page.

Creating Home.aspx

Start to create a content page, such as `Home.aspx`, from the Add New Item dialog box for an ASP.NET Web application. Notice the name Home for the page in Figure 11-15. VWDE will automatically add an `.aspx` extension to the filename. When you are creating a content page, you must also select the Select master page check box as Figure 11-15 indicates. When you click Add, VWDE does not immediately create a new Web form page. Instead, it opens the Select a Master Page dialog box. After you select a Master page and click OK, the page opens in VWDE.

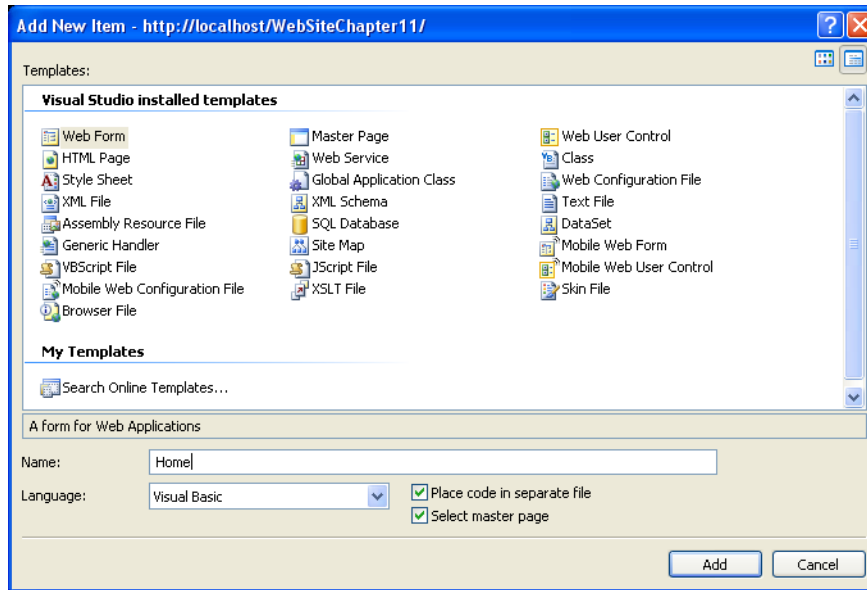


Figure 11-15. Create a content page by selecting the Select a master page check box.

After `Home.aspx` initially opens in VWDE, its Source tab shows both a `@Page` directive and a Content control. Figure 11-16 shows an excerpt from the Source tab within `Home.aspx` with the critical settings. Any content for a page belongs within the Content control. The `@Page` directive for a content page has a `MasterPageFile` attribute that references the Master page on which the content page is based. For example, the `Home.aspx` page in Figure 11-16 has an assignment of `~/Master1.master` for its `MasterPageFile` attribute. The Content control references the `ContentPlaceHolder1` tag in the `Master1.master` file. Notice how Figure 11-16 shows no additional content beyond the `@Page` directive and the Content control. This is because the initial formatting for the page depends on `Master1.master`.

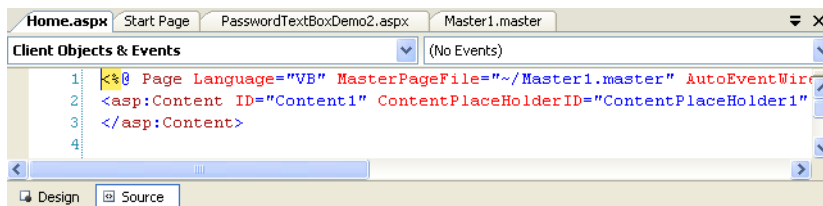


Figure 11-16. The initial settings for a new content page tie it back to a Master page.

You can enter text into a Content control from the Design tab. Your text is likely to appear within the opening and closing `span` tags if you have the View ~TRA Details menu command specified to show tags in the Design tab.

As mentioned a typical use for a home page is to provide hyperlinks for other pages within a Web site. You can add the hyperlinks from the Design tab by selecting some text, and then choosing Format ~TRA Convert to Hyperlink. Make sure the Type drop-down box is appropriate for the destination page, such as `http:` for a link to another page. If the destination page already exists within the Web site, then you can click Browse and navigate to the destination page for the hyperlink. Otherwise, you can type in the URL for the page. For example, the text on `Home.aspx` refers to the `About.aspx` page, which does not exist yet.

However, you can enter the URL that the page will have after it is created, such as `http://localhost/WebSiteChapter11/About.aspx`, in the Hyperlink dialog box.

You can preview the text on the page by choosing View in Browser from the context menu for `Home.aspx` within Solution Explorer. If you conclude the text appears too close to the top and left borders of the page, you can remedy this outcome with the following steps. An Offset from normal flow setting enables you specify top and left displacements of selected text from its normal position on a page.

- * Select the text and click the ellipsis button next to the Style property in the Properties Window. This opens the Style Builder dialog box.
- * Within the Style Builder dialog box, select the Position page.
- * Select Offset from normal flow in the Position mode drop-down box.
- * Then, insert a positive value, such as five, in the Top and Left boxes below the drop-down box. These settings offset the text from the page's top and left borders by 5 pixels (unless you choose another metric, such as millimeters). Pixels are the default metric for displacements.
- * After you click OK and return the focus to the Design tab, the opening span tag for the selected text converts to an anchor.

Note You should designate hyperlinks and do other formatting that requires selecting text before making Offset from normal flow settings. This is because the Offset from normal flow settings can disrupt your ability to select text within a span block in the Design tab.

Figure 11-17 shows a preview of the `Home.aspx` page with some sample text and the formatting operations for adding a hyperlink and offsetting the text. In addition, the font size setting is 14 point, which is larger than the default 12 point font size. Notice the slight offset of the text from the top and left borders of the page. You can adjust the size of this offset from the Style Builder dialog box or by adjusting the TOP and LEFT settings within the Style property for the `span` tags bounding the text in the Properties window. If you rest your cursor over the About page hyperlink, the URL for the hyperlink appears in the bottom border of the browser.

Figure 11-17 reflects the formatting of `Master1.master`. The top and bottom borders reflect the content that shows in Figure 11-14. The color of the side border in the content page within Figure 11-17 is gray as in the Master page depicted in Figure 11-14. These formatting features confirm that the content page inherits formatting from `Master1.master`.

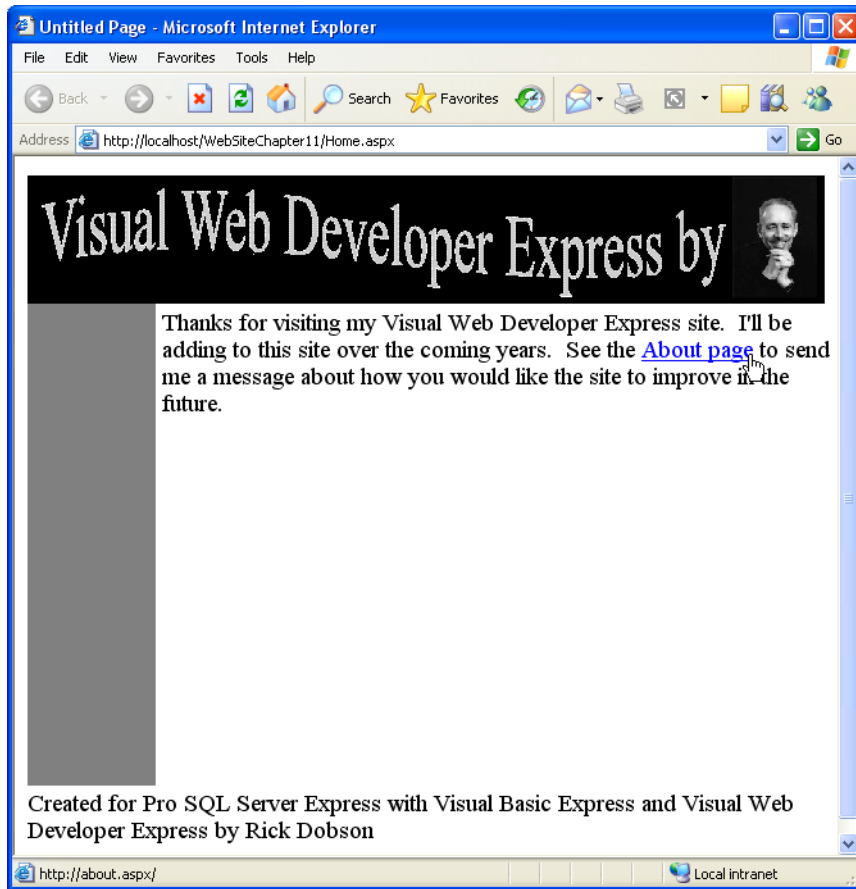


Figure 11-17. A content page can reflect formatting that you apply to the content page as well as settings that it inherits from its Master page.

Creating About.aspx

The `About.aspx` page thanks visitors for navigating to the Web site. In addition, this page provides a hyperlink for providing feedback through the local email client of the computer running the browser. A completed version of the `About.aspx` page appears in Figure 11-18. Clicking the hyperlink on the page opens the default email client, such as Microsoft Outlook, on the computer running the browser session. Figure 11-19 depicts an open Outlook session with the To and Subject fields populated. All the user has to do is enter a short note. In the process of collecting feedback via an email client, you automatically collect the email address of the sender, which provides you a means to start building a relationship with the Web site visitor.

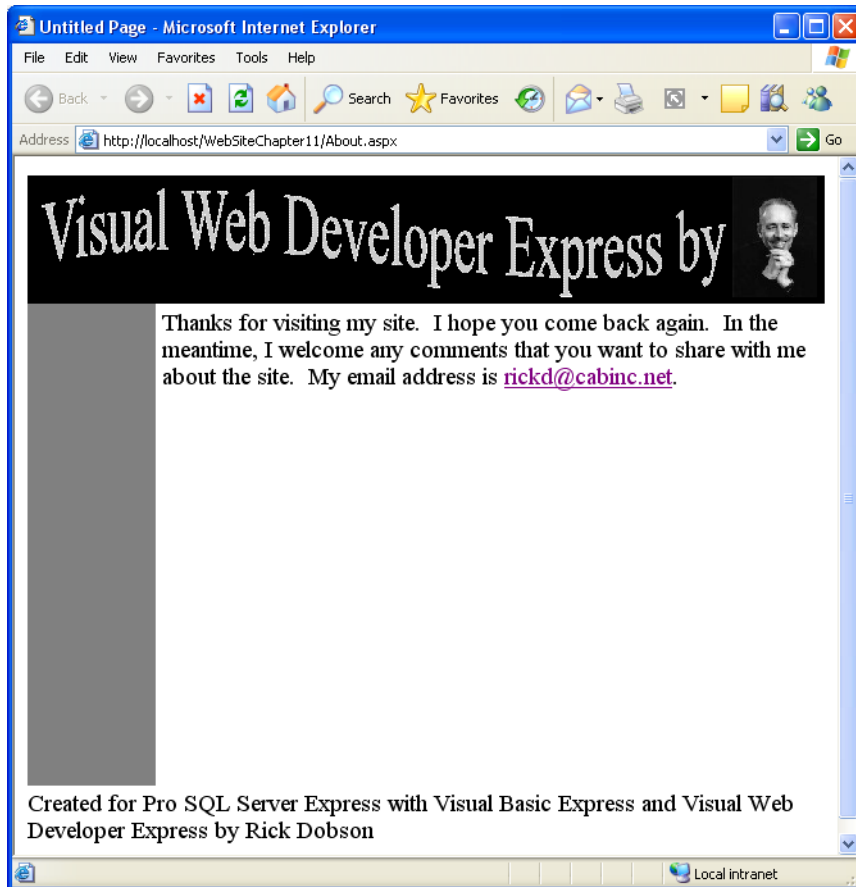


Figure 11-18. You can write a content page to open an email client and send a reply back to the Webmaster.

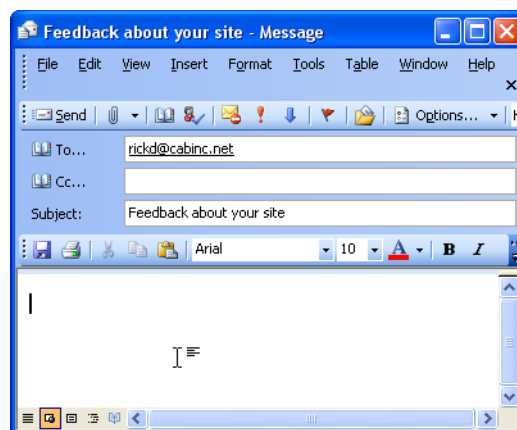


Figure 11-19. You can populate To and Subject fields for an email message from a hyperlink on a content page.

VWDE does not by default recognize an email address in a stream of characters as a hyperlink. As a consequence, VWDE does not automatically format email addresses as

hyperlinks. Therefore, you have to manually add the hyperlink to an email address if you prefer to have the email address serve as a hyperlink that can automatically open the default email client on a computer as in Figure 11-18 and Figure 11-19. You can manually add the hyperlink as follows.

- * Select the email address (rickd@cabinc.net) on the content page.
- * Choose Format ~TRA Convert to Hyperlink to open the Hyperlink dialog box.
- * Select mailto: in the Type drop-down box.
- * Trail the box for populating the `mailto:` attribute with the following text:
`rickd@cabinc.net?subject=Feedback about your site.`

You may recognize the syntax for specifying the subject field in the email message as standard format for designating a querystring parameter after a URL. You can specify additional email message fields, such as body, cc, and bcc, besides subject. Place a preceding ampersand (&) before each additional field after the first one following the question mark (?), and then specify the field with the same format as the first field. For example, the following `mailto:` attribute specification designates body and cc fields after the subject field.

`mailto:name@domain.com?subject=short&body=not long&cc:other@domain.com`

Converting a User Control Solution

It is likely that you will encounter a situation where it is desirable to convert an existing page for use with a Master page. The `PasswordTextBoxDemo2.aspx` page illustrates an interesting page for conversion because it is relatively simple, but it contains a user control. Because you need to register a user control on a page before you can use it, converting a page with a user control is slightly more complicated than converting one without a user control. However, since the `PasswordTextBoxDemo2.aspx` page contains several standard Web form controls, you'll learn how to handle built-in controls as well.

There are three main steps to the conversion of the page.

- * First, you need a new Web form page based on your Master page. You can create a new page named `PasswordTextBoxDemo3.aspx` based on the `Master1.master` page.
- * Second, open the Design tab of the page that you want to convert (`PasswordTextBoxDemo2.aspx`). Then, copy all its controls to the Design tab of the new content page (`PasswordTextBoxDemo3.aspx`).
 - * The copy attempt will fail for the `PasswordTextBox` user control, but it will succeed for all other controls on the page. See Figure 11-20.
 - * Select the error notice and remove it after the copy completes.
 - * Then, drag the `PasswordTextBox` user control to `PasswordTextBoxDemo3.aspx` as demonstrated in Figure 11-4. This dragging process automatically registers the user control on the `PasswordTextBoxDemo3.aspx` content page.
- * Third, you can copy the code from the `Button2_Click` procedure in the module for `PasswordTextBoxDemo2.aspx.vb` to the module in `PasswordTextBoxDemo3.aspx.vb`.

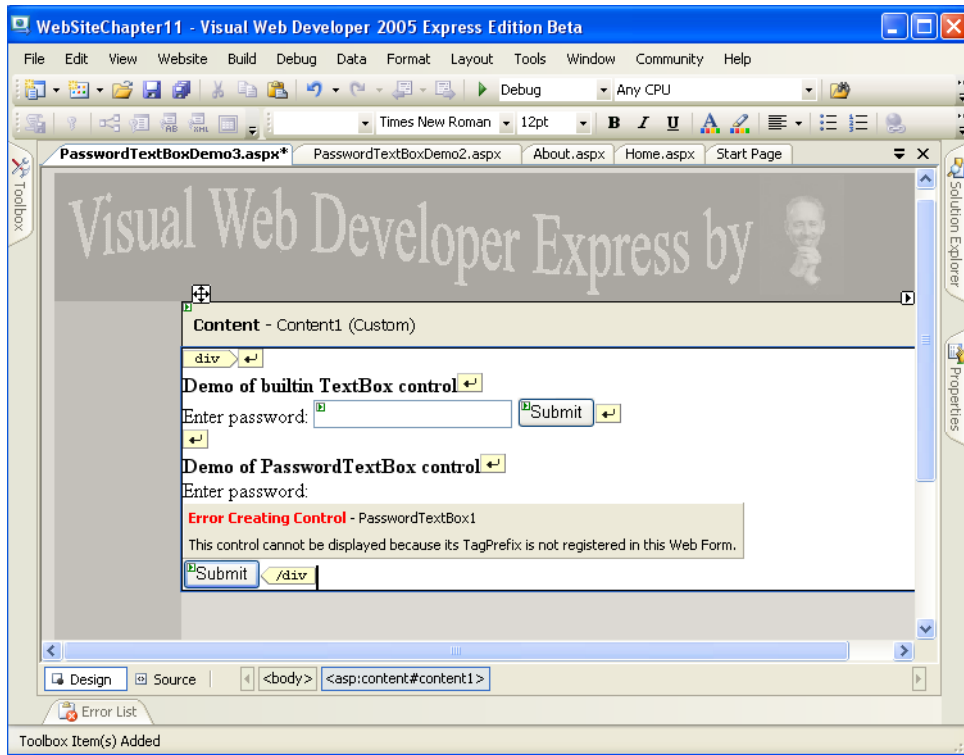


Figure 11-20. You can copy built-in controls from a Web form page to a content page, but you must register (for example, by dragging) a user control on a content page.

Figure 11-21 shows the converted solution with the same password typed in the top and second TextBox controls. The second TextBox control is actually the `PasswordTextBox` user control. When a user clicks the Submit button next to the user control, the page makes a round trip to the server and writes Password OK above the page's top border. This style of feedback defeats the elegant appearance of the content page based on `Master1.master` by entering content outside the bounds of the Master page.

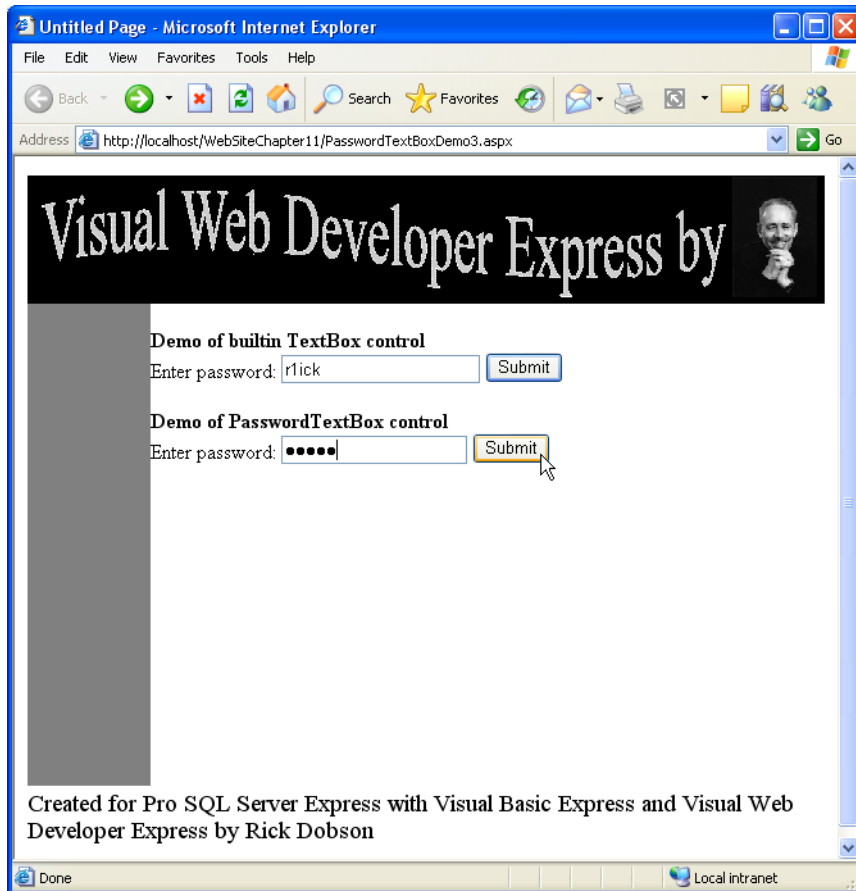


Figure 11-21. PasswordTextBoxDemo3.aspx has the same functionality as PasswordTextBoxDem2.aspx, except that the new version (PasswordTextBoxDemo3.aspx) has a more professional look based on Master1.master.

I am including in the sample files for this chapter another page (PasswordTextBoxDemo4.aspx) that is an update of PasswordTextBoxDemo3.aspx. The updated page writes the feedback about the validity of the password to a Label control below the other controls on the Web form. This solution is superior for a content page because it does not write outside the bounds of a Master page as PasswordTextBoxDemo3.aspx does. Although PasswordTextBoxDemo4.aspx does not appear in this section, it is available with the files for the WebsiteChapter11 Web site. In addition, the next section references the PasswordTextBoxDemo4.aspx file.

Adding and Using a Menu for Navigation

The WebsiteChapter11 Web site has four content pages: Home.aspx, About.aspx, and PasswordTextBoxDemo3.aspx as well as PasswordTextBoxDemo4.aspx. This is a very small number of pages for a Web site. It is not unusual for sites that have been in operation for any significant amount of time to have hundreds of pages. As the number of pages at your site grows, it is often desirable to have a menu to help site visitors navigate to pages on your site that they want to view.

VWDE features a Menu control in the Navigation group of its Toolbox. This Menu control lets you create a menu for navigation to pages at your site or even other sites without you having to write any code. This section demonstrates a straightforward application of the Menu control to achieve both of these goals. While you can place a Menu control on any page in a Web site, it often makes sense to place a Menu control on a Master page. This means the menu is available from all content pages based on the Master page with the Menu control.

You can add a Menu control to a page by dragging it from the Navigation group of the Toolbox to a Web site page. The Menu control displays with an Orientation property setting of Vertical or Horizontal. With the Menu control selected, you can set the Orientation property from the Properties window. If you drag the Menu control to the top border of a page, use the Horizontal setting for the Orientation property. In a side area, a Vertical setting for the Orientation property is more likely to make the Menu control display better. Because the Menu control is such a rich one, there are many other properties, besides Orientation, that you can set from the Properties window for the control. The sample for this section uses a couple of formatting properties from the Properties window to modify the default formatting for menu items.

Drag a Menu control to the side area on the `Master1.master` page in the `WebSiteChapter11` Web site. Set its Orientation property to Vertical (if the Menu control does not already appear with a vertical orientation). From the control's smart tag menu items, click Edit Menu Items to open the Menu Item Editor dialog box. Recall that the "Using a Calendar Control" section in Chapter 10 previously described the use of smart tags.

Figure 11-22 shows the Menu Item Editor for `Master1.master` after all the items are added to the menu for the Master page. As you can see, the editor has two boxes.

- * The Items box on the left lets you add, nest, and delete menu items. Its toolbar controls from left to right enable the following actions.
 - * Add a root item – lets you add menu items such as Home, PasswordTextBoxDemos, About, and Favorites
 - * Add a child item – lets you add an item nested below the currently selected item
 - * Remove an item – lets you drop the currently selected item and its nested items
 - * Move Up – lets you move an item up among its siblings
 - * Move Down – lets you move an item down among its siblings
 - * Make Sibling – lets you make the selected item a sibling of its parent
 - * Make Child – lets you make the selected item a child of its preceding sibling
- * The Properties box on the right lets you assign settings for a selected menu item in the Items box.
 - * When you initially add a new item, it will have the Text property of New Item. You can assign a more meaningful Text property by entering a new value, such as Home, in the Properties box.
 - * Use the NavigateURL property to designate the page to which a menu item transfers control. An ellipsis button (see Figure 11-22) lets you designate pages within the current site by navigating to them. Type or copy the URL from the

Address box of a browser for a page outside the site. The three child menu items below Favorites have destination URLs outside the WebSiteChapter11 Web site.

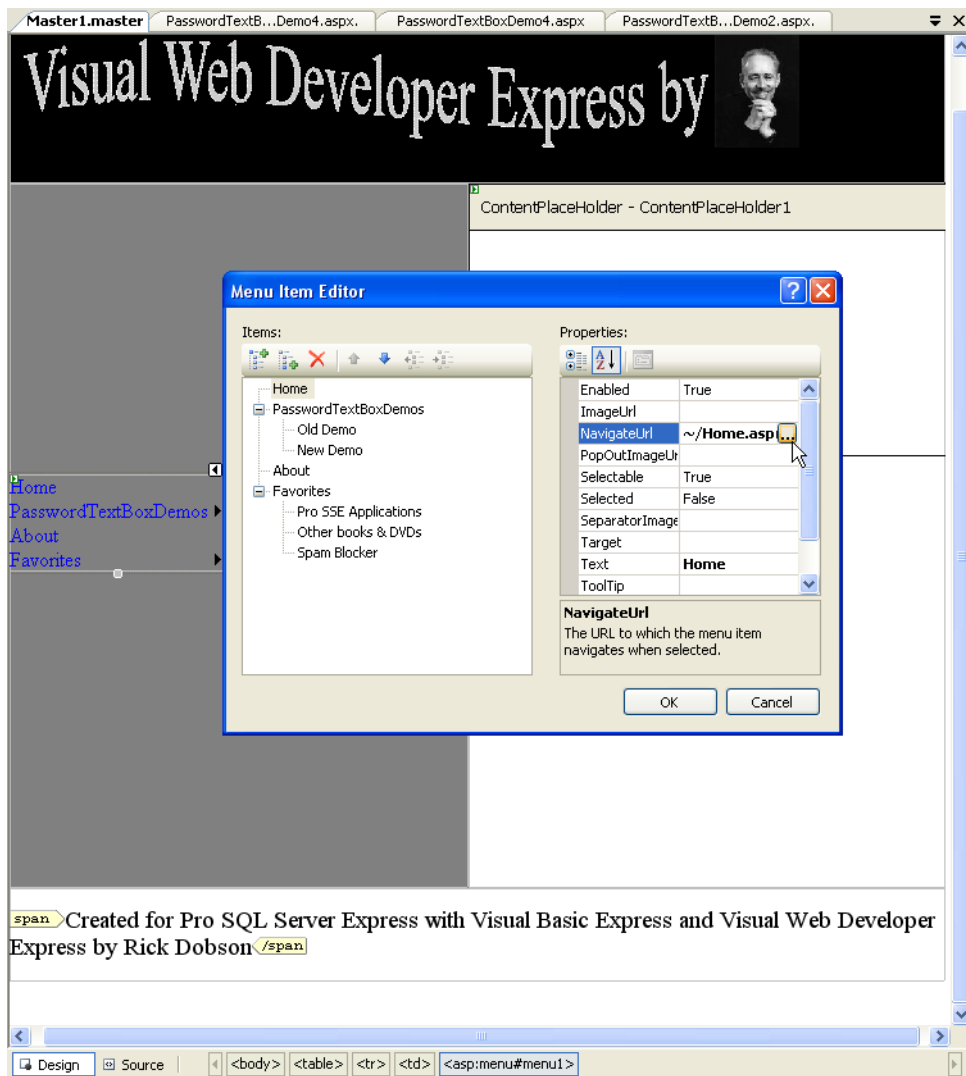


Figure 11-22. Use the Menu Item Editor to specify the names of pages to which visitors can navigate from a Menu control.

After you add menu items and specify the pages to which they should transfer control, you may want to perform some additional control editing. Notice the item Text property values are not prominent against the gray color for the page's side border in Figure 11-22. To remedy this situation, you can select the Menu control and edit its properties in the Properties window. Expand StaticMenuItemStyle, and then from within the StaticMenuItemStyle properties expand the Font menu of properties. Change the static menu item's ForeColor property to White (a multi-tabbed list box helps you perform the task). In addition, update the font's Bold property from False to True. Figure 11-23 shows the Properties window after the selection of White as the font color and the process of setting the Bold property to True. You can perform similar editing for the child menu items with the

DynamicMenuItemStyle group of properties. This example assigns a ForeColor of Red in a bold font to child menu items.

Tip As you make changes to Web pages and then preview them by choosing View in Browser from their context menu, you may notice that your last set of changes is not rendered by your browser. One possible source of the problem may be that VWDE did not automatically update the page. Another possibility is that the browser is showing a cached version of the page before you made your changes. You can remedy either problem with the following steps. First, close the browser. Then, choose the Save control on the Standard toolbar in VWDE to ensure your changes are committed. Finally, re-open the page with the View in Browser context menu command. If these steps fail to render a change you believe you made, consider performing the edits over again. Also, make sure that your browser is not working with a previously cached version of the page.

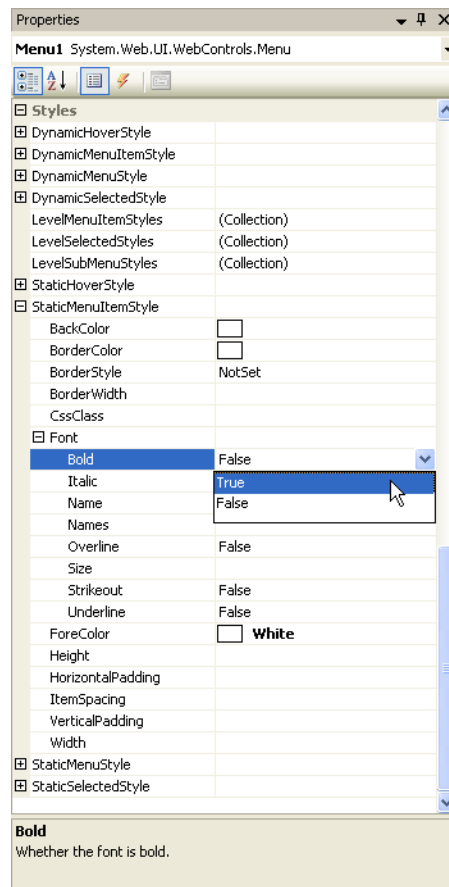


Figure 11-23. The Properties window lets you change how items appear in a Menu control.

Figure 11-24 shows Home.aspx after the addition and editing of the Menu control on Master1.master. Child menu items are showing because the mouse hovered over the PasswordTextBoxDemos menu item. After the child items appeared, I moved the mouse to the New Demo child item. The menu items are much easier to read because the main menu items appear in a bold, white font and the child items appear in a bold, red font. The more

graphically able readers are probably thinking how much better they could make the main and child menu items look. My reply is that you can, and that it is easy to do with the Properties window for a Menu control.

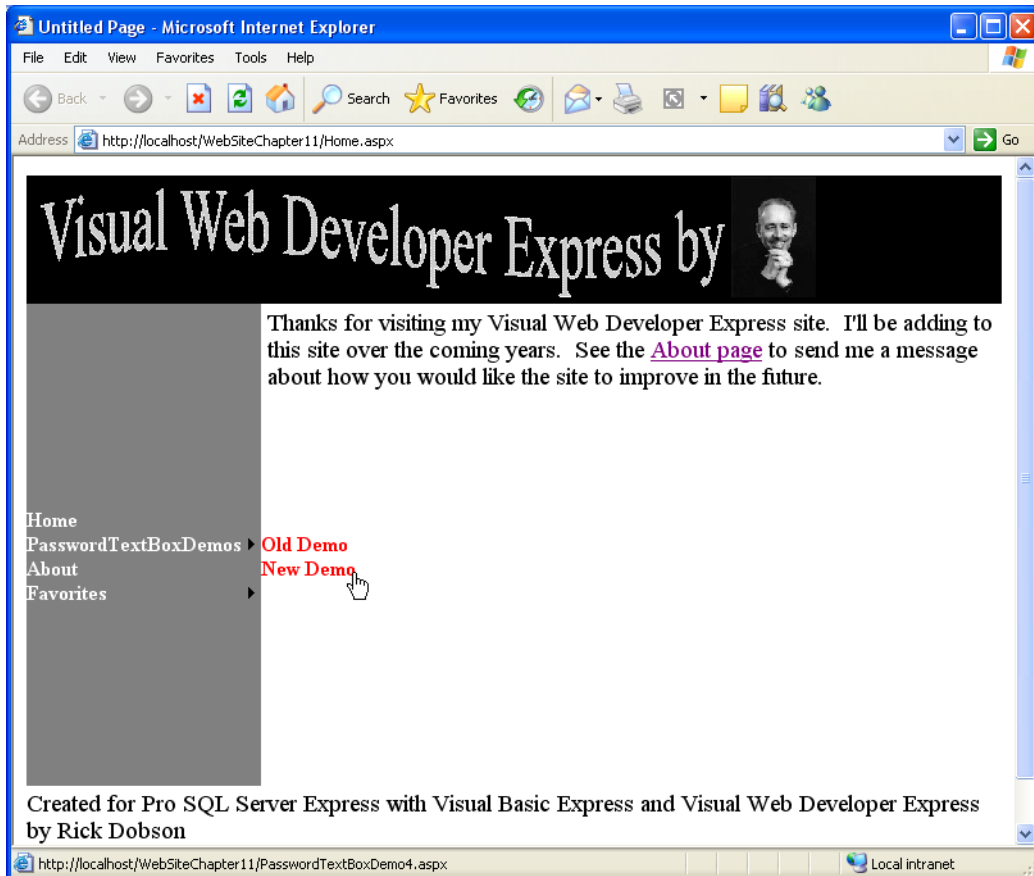


Figure 11-24. A Menu control makes it easy to navigate across the page in a Web site.

Login Controls and Membership Features

A set of Login controls facilitates a new set of built-in membership features for VWDE. Allowing users to log in to a Web site is growing in popularity. With the capability to make users log in and track them based on their login settings, Web sites can make access to some or all its resources conditional on a user's identity. You can even group logged-in users into roles, and control access to resources based on user membership in roles.

The Login controls allow you to construct a login solution with no code at all. Furthermore, you can customize and extend the built-in functionality available from these controls through code or even by making property value assignments at design time. The Login controls implement a membership model because those who authenticate themselves at a Web site can gain an elevated set of permissions relative to those who log in anonymously. In a sense, the logged-in users are members.

One of the nice features of the Login controls is that enabling users to log in does not force you to restrict access to just those who do so. Users can still connect to a Web site anonymously without identifying themselves. Therefore, implementing a membership model

with the Login controls does not force you to restrict access to a site only to those who want to or can identify themselves.

Note When users connect to a Web site anonymously, which is by far the most common way to connect to most Internet sites, the Web site does not keep track of who the users are. Therefore, the Web site has no way of granting special permissions to users based on their identity.

Although the Login controls require no code for their use, they implement a very rich feature set. Some of the most readily available features enable you to:

- * Let users log in to a Web site with a user name and password over the Internet without requiring a Windows login.
- * Let users register themselves as members, provide membership registration through a Web site administrator, or permit both styles of user registration concurrently.
- * Determine the content that users view on a page so that logged-in users get one view of a page and anonymous users get a different view of the same page.
- * Grant or deny access rights to Web site folders based on user identity and user membership in roles. All users connecting anonymously share a single, common identity – namely, that for the anonymous user.
- * Let users change their password and recover their passwords if they forget it.
- * Provide feedback about the login status of a user so that a user can confirm if she is logged in and what user name she is using. The ability to connect in different ways and get feedback about how you are connected is especially convenient for developers who need to test a login solution.

Seven individual Login controls are available to help you manage the above feature set without any code. You can drag these controls to a Web form page from the Toolbox in VWDE. The controls reside in a Login group within the Toolbox. A brief summary of each control follows. Controls appear in their order within the Login group on the Toolbox.

- * A Login control lets a user log in by user name and password.
- * A LoginView control lets you specify the content that a user sees on a page based on their login status and membership in roles.
- * PasswordRecovery is actually a misnomer because users cannot recover their existing password. Instead, the control creates a new password for the user and mails the password to the user.
- * LoginStatus provides two capabilities; these allow you to:
 - * Detect whether the current user is anonymously connected or authenticated by a user name
 - * Toggle a hyperlink to log in or to log out of a Web site

- * `LoginName` returns the name of the current user. This control is particularly convenient for customizing a greeting on a Web form page.
- * `CreateUserWizard` helps you to create a Web form for user self-registration; as a minimum the form contains 2 fields: user name and password. If you implement password recovery, you can use the wizard to create three additional fields: email address, question for recovery, and answer to question for recovery. You can optionally specify the collection of additional custom fields.
- * `ChangePassword` provides a user interface for automating the updating of a password by a user. Users can also adapt this control to assign a meaningful password that is easier to remember than the one mailed after the password-recovery process.

The VWDE Help system includes several walkthroughs that demonstrate the operation of Login controls. My favorite VWDE Help page has the title “Walkthrough: Creating a Web Site with Membership and User Logins (Visual Studio)”. This section in this chapter of the book complements the built-in documentation in three ways.

- * First, this section drills down on selected setup features to help you configure a Web site to use the Login controls.
- * Second, this section demonstrates selected features – particularly password recovery – in more depth than the built-in documentation.
- * Third, you will learn simple programming techniques for extending and customizing the built-in controls.

Configuring a Web Site and a Web Server for Login Controls

Before you can use the Login controls, you need to configure your Web site to run them. The easiest way to do this is with the ASP.NET Web Site Administration Tool. This tool has three main tabs and a Home tab that provides information about the role of each tab. Using the tool will allow you to complete three tasks that let you use the Login controls. These tasks are to:

- * Set up a provider for storing membership and role data
- * Create some initial Web site members and optionally some groups
- * Designate a SMTP server that supports sending passwords to user email accounts

You will use the ASP.NET Web Site Administration Tool to perform selected other functions when you invoke the Security Setup Wizard. In addition, you can return to the ASP.NET Web Site Administration Tool after initial configuration to add more users and roles, assign users to roles, create access rules, and perform other administrative tasks for your Web site. For example, you can designate a SMTP server that supports sending information to users. You can perform this task before or after running the Security Setup Wizard.

Running the Security Setup Wizard

Launch the ASP.NET Web Site Administration Tool by choosing Website ~TRA ASP.NET Configuration from the Standard VWDE menu. The tool opens to its Home tab. Next, start the Security Setup Wizard. Click the Security tab, and then click the link labeled: Use the security Setup Wizard to configure security step by step.

The Welcome Screen is the first of seven screens in the wizard. It provides an overview of the wizard's functionality. You can move on to the first screen for making a setting by clicking Next in the lower-right corner of the screen. This screen allows you to specify an access method for logins. Unless you want to restrict access to users who have Windows accounts, choose the radio button labeled From the Internet. This selection is likely to be the typical one for most users of the Login controls. For example, if you choose the radio button labeled From a local area network, there is no need for a Login control because the Web site will accept your Windows login.

The next screen allows you to make a selection about the type of data provider or informs you of the selection made to store security settings. When you install SQL Server Express as described in Chapter 1, the wizard automatically uses a data provider for SQL Server Express. As a result, the wizard adds a SQL Server Express database to the APP_Data folder of the site.

Tip Install SQL Server Express on a computer before running the Security Setup Wizard. This preliminary step allows the Security Setup Wizard to automatically use SQL Server Express to automatically create a SQL Server Express database that stores membership and security settings.

The next screen allows you to define roles for grouping users. You can also use roles for assigning access permissions to a Web site and the folders within it. If you do not set up roles when using the Security Setup Wizard, you can open the ASP.NET Web Site Administration Tool and specify roles later.

The next screen allows you to add new users. Figure 11-25 shows the screen for a new user account for a user named Rick Dobson. You may need to tweak your password and make it more complicated than you are used to because default settings require a complex password. A complex password is often called a strong password because it is more resistant to hacking than a simpler password. A mixture of lowercase, uppercase, and numeric characters along with special characters (for example, ! or _) is sufficient to satisfy the requirement for a strong password. There is also a length requirement of at least seven characters. After you click Create User, you are given the opportunity to continue creating more users. I also created two other user accounts with names of Virginia Dobson and Tony Hill for testing purposes. When you finish creating users, choose Next.

The screenshot shows the 'Security Setup Wizard' interface. On the left, a blue sidebar lists steps 1 through 7, with 'Step 5: Add New Users' highlighted. The main content area has a blue header 'Create User' and text explaining that users can be added by entering ID, password, and email, and optionally a security question. Below this is a form titled 'Sign Up for Your New Account' with fields for User Name (Rick Dobson), Password (masked), Confirm Password (masked), E-mail (rickd@cabinc.net), Security Question (wife's name?), and Security Answer (Virginia). A 'Create User' button is at the bottom right of the form. Below the form is a checkbox labeled 'Active User' which is checked. At the bottom of the wizard are three buttons: '< Back', 'Next >', and 'Finish'.

ASP.NET Web Site Administration Tool

Security Setup Wizard

Step 1: Welcome
Step 2: Select Access Method
Step 3: Data Store
Step 4: Define Roles
Step 5: Add New Users
Step 6: Add New Access Rules
Step 7: Complete

Add a user by entering the user's ID, password, and e-mail on this page. You can also specify a question with an answer that the user must give when resetting a password or requesting a forgotten password.

Create User

Sign Up for Your New Account

User Name: Rick Dobson

Password:

Confirm Password:

E-mail: rickd@cabinc.net

Security Question: wife's name?

Security Answer: Virginia

Create User

☒ Active User

Security Management

< Back Next > Finish

Figure 11-25. You can create one or more users from inside the Security Setup Wizard.

The next screen allows you to designate access rules. Recall that access rules apply to folders within a Web site. Therefore, you will typically want to defer setting access rules until you create more folders. It is common to deny access to selected folders to anonymous users. You can also allow or deny access to folders to individual users or roles.

The final screen has the label Complete and it announces that you have successfully completed the Security Setup Wizard. In repeated runs of this Wizard for different Web sites, I always encountered success. Click Finish to exit the wizard and return to the ASP.NET Web Site Administration Tool.

Figure 11-26 shows the Security tab of the ASP.NET Web Site Administration Tool. The screen has three panels with links within each panel. The left panel helps you to manage users. The second panel helps you to manage roles, and the third panel is useful for managing folder access rules.

Notice that the Users panel on the left specifies that three users exist. These correspond to Rick Dobson, Virginia Dobson, and Tony Hill. You can use the Manage Users link to add or remove users. In addition, if you had roles defined, you can add or remove users from membership in roles. Whether or not a user is a member of a custom role, a user is a

registered member of the Web site. This status is distinctly different than anonymous users who can typically connect to a Web site, but who are not registered Web site members.

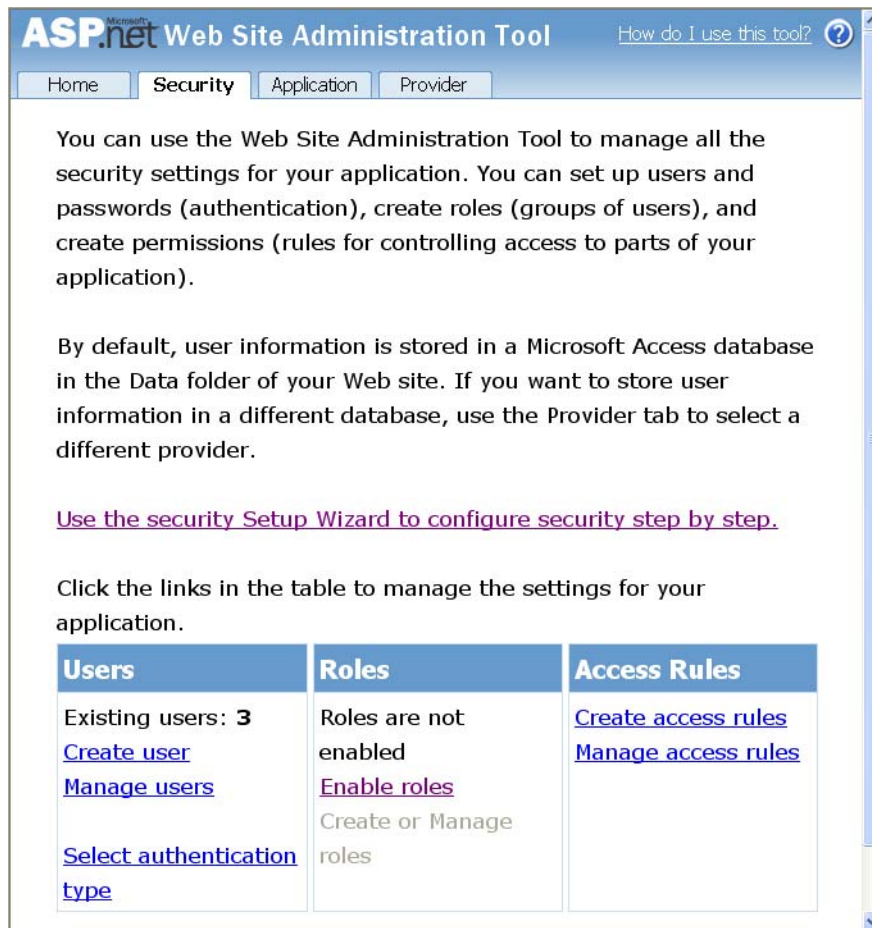


Figure 11-26. You can manage users, roles, and access rules from the Security panel of the ASP.NET Web Site Administration Tool.

You should experiment with all three panels, but it is not necessary to do so for the Login control examples presented in this book. In general, roles will become more valuable as the number of users rises to more than three so that you have several users in each of several different roles. Because access rules apply to folders within a Web site, you should at least have one other folder besides one of the built-in folders before you begin to specify access rules. The last Login controls example is a rich one that includes coverage of access rules along with other features for Login controls.

Configuring SMTP E-mail Settings

Click the Application tab in the ASP.NET Web Site Administration Tool to enable the password recovery feature. The PasswordRecovery control depends on the availability of a SMTP server. Click the link labeled Configure SMTP e-mail settings to open a Web page with a Configure a SMTP section for designating a SMTP server that your Web site will use to mail passwords to users.

There are two critical settings for configuring a Web site for use with a SMTP server. First, you designate a server. Use the Server Name box in the Configure a SMTP section of the Web page for this. Second, you specify an e-mail address for the From field of email messages from the Web site. Use the From box in the Configure a SMTP section of the Web page for this. The default port for an SMTP server is 25. Do not change the default port unless you have information about another port being appropriate from a SMTP administrator.

Note If you are not familiar with SMTP servers, but you use an email client, such as Microsoft Outlook, you will be able to discover suitable settings from your email client to configure your Web site for use with a SMTP server. For example, choose Tools ~TRA Email Accounts from the Outlook Standard menu. Select the View or change existing e-mail accounts radio button and click Next. Select an e-mail account from which you wish to send e-mail from your Web site and click Change (you will not actually be changing the settings – you just want to view the settings for the account). The entry for Server Name when you configure your Web site for use with a SMTP server is the entry in the Outgoing mail server (SMTP) box of the Outlook dialog box. The entry for From when you configure your Web site for use with a SMTP server is the entry in the User name box of the Outlook dialog box.

Granting Access by User Name and Password

Clearly the first step to tracking users by their login is to open a form that lets users input a user name and password, and then associating the user inside the application by their user name. These basic kinds of tasks are so common that there should be a highly simplified and standardized way of performing them. The virtues of standardization are considerable. By using standard techniques, such as those implemented by the Login controls, you are sure that a basic task will always work one way. Even if it does become necessary to customize a standard technique at least the customized implementation relies on standard technique as a starting point.

The Login control from the Login group within the Toolbox lets you add a standard form to a Web page with a component behind it. This component is part of the ASP.NET Membership class. By default, ASP.NET expects the Login control to reside on a page named `Login.aspx` (although you can override this from the `web.config` file in a Web site's root folder). Also, the Login control transfers control to a destination URL. By default, the destination URL after a successful login is `default.aspx` in a Web site's root folder, but you can override this setting by assigning a String value to the control's `DestinationURL` property with the path to another file.

The following Note explains a trick used with the `WebSiteChapter11` Web site. The application of the trick is desirable because any one Web site can have just one page named `default.aspx`, but it is convenient for demonstrating the Login controls to have multiple `default.aspx` pages. The trick essentially allows you to re-configure the sole `default.aspx` page at a Web site by re-writing the HTML code, including its Membership class controls, for the page. This trick facilitates demonstrating the usage of different Login controls on the single `default.aspx` page at a Web site.

Note The WebSiteChapter11 Web site includes the text for several `default.aspx` pages within it. Each version illustrates different features. The text filenames are `default1.txt`, `default2.txt`, and `default3.aspx`. The `default2.txt` file also has a matching `default2vb.txt` file for an associated code file (`default.aspx.vb`). The version of `default.aspx` for this section is from the `default1.txt` file. You can run the sample for this section by copying the text in `default1.txt` over the contents of the Source tab in `default.aspx`.

As a result of these default settings, you will typically start to use the Login control by creating a new file named `Login.aspx` for the Login page at a site. You can then drag the Login control from the Toolbox to the page. In the event of faulty input, the Login control will provide some feedback to help a user correctly input values to the form on the `Login.aspx` page. However, the Login control, like several other Login controls, can integrate tightly with the `ValidationSummary` control to provide richer feedback about input errors. The `ValidationSummary` control is one of the built-in Validation controls in the Validation group within the Toolbox.

Figure 11-27 shows a Login control above a `ValidationSummary` control in the WebSiteChapter11 Web site. You can use the smart tag actions to customize the look of the control to one of a handful of `AutoFormat` settings. Alternatively, you can more granularly specify the appearance and behavior of the Login control with the Properties window. The excerpt from the Properties window shows properties for setting the Text property for controls nested within the Login control as well as the color, style, width, and padding of the border around the edge of the Login control. The selected property, `DestinationPageURL`, allows a page author to override `default.aspx` for the page to view after a successful login. In addition to setting these properties from the Properties window, you can also set them programmatically in the `Load` event procedure for the Login page.

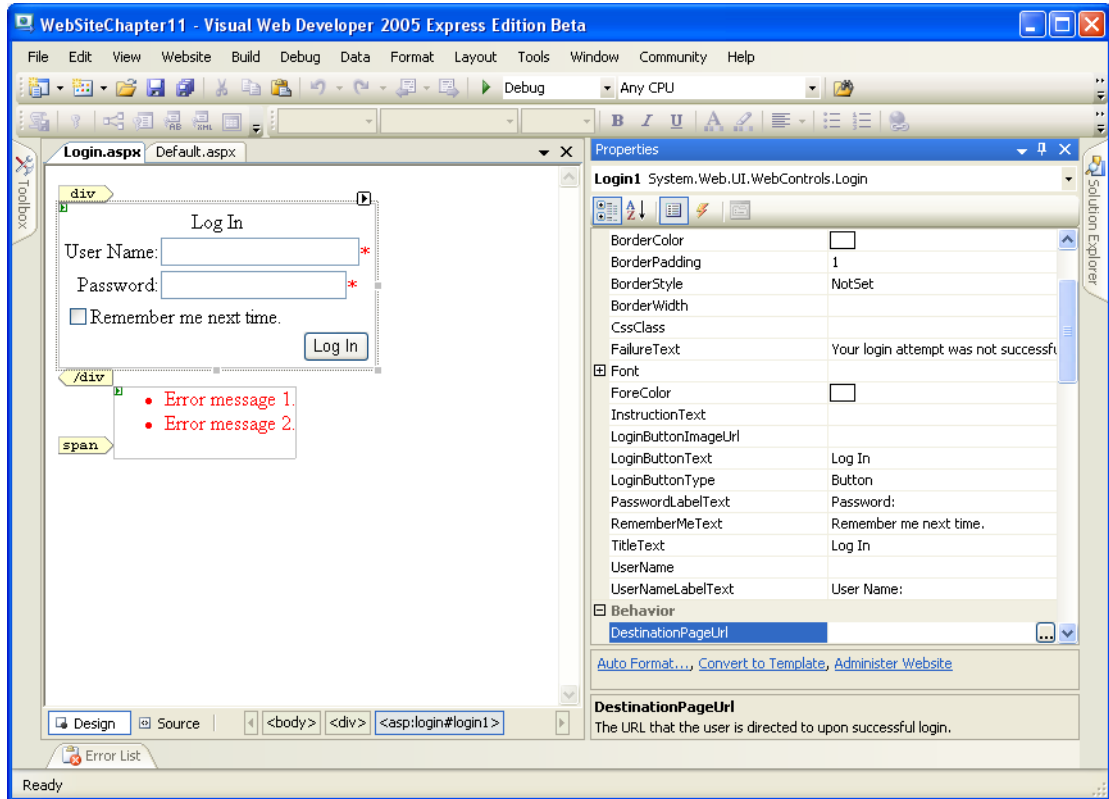


Figure 11-27. The appearance and operation of the Login control is highly customizable with the help of the Properties window.

Figure 11-28 shows the Login control in a browser window with a valid user name and password. Clicking the Log In button transfers control to **default.aspx** unless the Login control's **DestinationURL** property specifies an alternate location for transfer after a successful login.

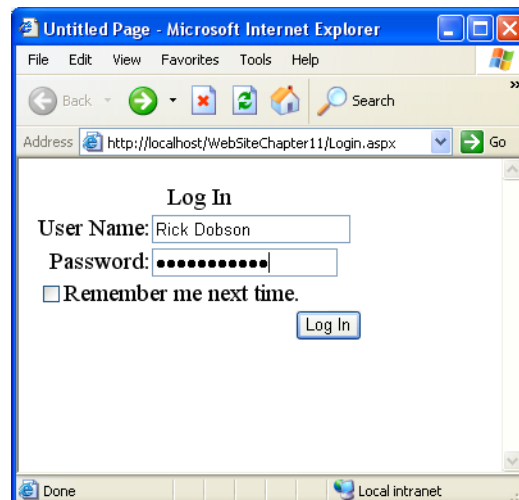


Figure 11-28. The Login control accepts a User name and password to identify the user for a session, and transfers control to a destination URL, such as `default.aspx`.

If a user fails to input either her user name or password, the Login control reminds the user with a red asterisk next to the control with the missing input. If you have a `ValidationSummary` control below the Login control, the control will additionally display text messages explaining what input is needed for the control to operate properly. If a user inputs either an invalid user name or password, the Login control returns a message saying the attempt was not successful. At that point, a user can try again with a different user name or password.

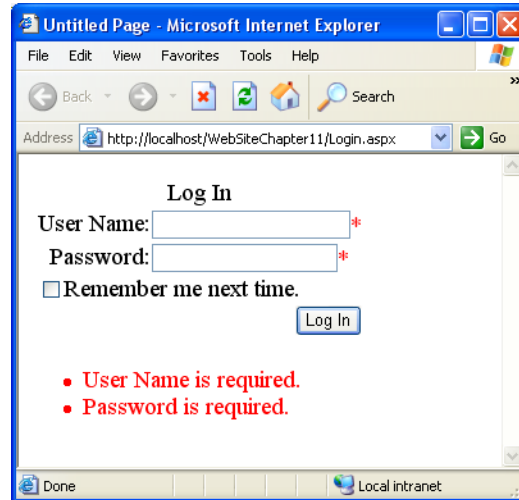


Figure 11-29. The Login control provides a range of messages to inform users of input errors associated with its use.

One of the benefits of having users log in is that you can customize their Web site experience. For example, Figure 11-30 shows the `default.aspx` page after it receives focus following the successful login attempt with the input appearing in Figure 11-28. This page includes two Login controls that automatically customize the look of a page depending on the user.

- * Notice that the heading on the page greets the user by their name. A `LoginName` control facilitates this by returning the name of the session's current user. If a user goes to `default.aspx` without first logging in, the user has an anonymous login which corresponds to an empty string value for the `LoginName` control.
- * The `default.aspx` page also includes a link with text of Logout. If a user clicks the link, VWDE logs out the current user and returns control to `Login.aspx`. A `LoginStatus` control provides this link. The `LoginStatus` control is another of the special Login controls. If a user navigates directly to `default.aspx` without first logging in, then the text for the link reads Login. Clicking the Login link transfers control to the Login page so that an anonymous user can log in with a known user name.

Note The terms Logout and Login are default terms for the links that the LoginStatus control shows. You can override the default link terms, but I recommend against it unless you have a compelling reason. Every change that you do not make is one less chance for an error to creep into your solution. Furthermore, standardizing on the default behavior makes it easier for your ASP.NET solutions to have a consistent look and feel for users.

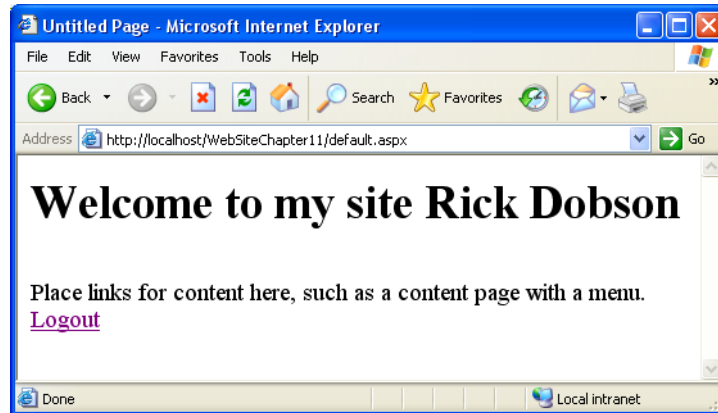


Figure 11-30. Login controls can help you customize pages based on a user's name and login status.

Figure 11-31 shows `default.aspx` from its Design tab with the `LoginName` control selected. The `LoginName` control returns the same `String` value (with a blank space after it) as the `Name` property of the `My.User` object; this `String` value is the name of the user (with a trailing blank), such as `Rick Dobson` . However, the `LoginName` control contains a long list of properties that enables you to format its appearance. The Properties window in Figure 11-31 shows a subset of the properties for the `LoginName` control. The `LoginName` control is within opening and closing `h1` tags that give it a bold font along with the rest of a heading at the top of the page.

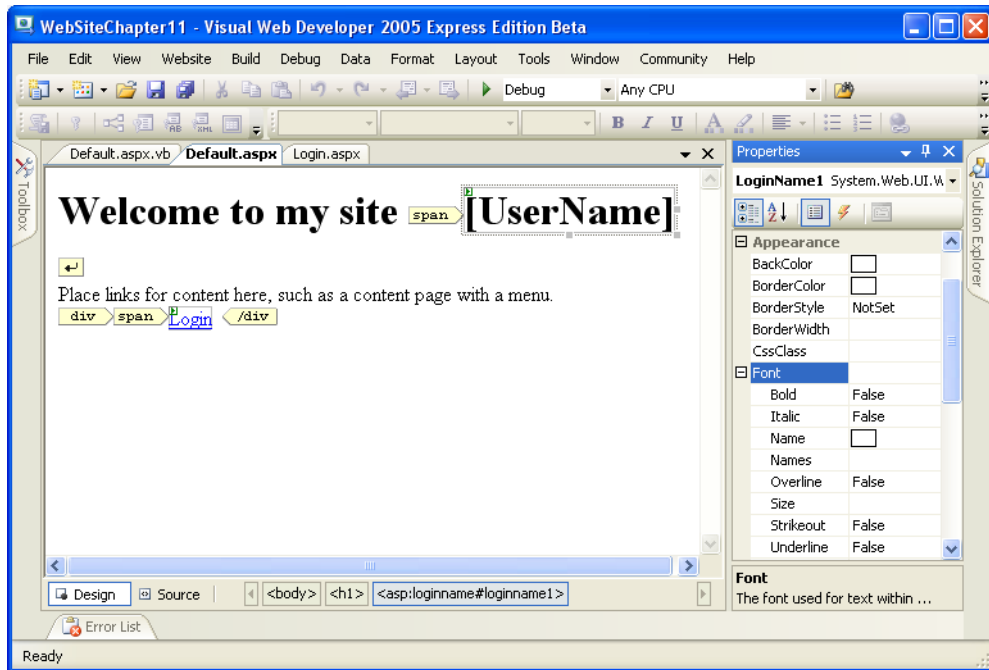


Figure 11-31. Login controls can help you customize pages based on a user's name and login status.

Tip If you have users from multiple computers logging into a Web site running on a Windows XP system, it is likely that you will have to make some adjustment to the Windows XP firewall to allow users to connect to the input. When I was testing the Login control features, I created an exception to the firewall for connection to port 80, the default port for an IIS Web server, from computers on my office's intranet. The exception that's right for you will depend on your computing environment and how you plan on using a Web site.

Customizing the Operation of Login Controls

The sample for this section demonstrates more advanced customization options than those covered in the preceding section. Special focus goes to contrasting customizations for registered users versus anonymous users. The sample application uses five pages. The `Login.aspx` page is the same as in the preceding sample. However, the `default.aspx` page is brand new. This new page features a `LoginView` control that makes it easy to customize the content showing to logged-in users versus anonymous users. In addition, `default.aspx` contains an ASP.NET Hyperlink control below the `LoginView` control. Code behind the page assigns the `Text` and `NavigateURL` property values based on whether the user is logged in with a user name or connected to the Web site as an anonymous user.

Note To run the sample for this section, you need to copy the text in `default2.txt` over the content in the Source tab of `default.aspx`. After that, open the code page behind `default.aspx` by choosing View Code from the context menu for the page. Then, copy the text in `default2vb.txt` over the content in the module

within `default.aspx.vb`.

Aside from the Login and default pages, the sample application includes three more pages.

- * `MemberPage.aspx` is available only to logged-in users. Code in the Load event procedure behind this page redirects users who are not logged in to another page (`VisitorPage.aspx`).
- * `VisitorPage.aspx` is available to any user of the Web site – whether they are logged in or not.
- * `LoginStatus.aspx` offers a route for logged-in users to log in as a new user. If an anonymous user encounters this page, the user is invited to log in. Code behind this page formats controls dynamically depending on whether a logged-in user or an anonymous user views it.

Making the Content on a Page Dynamic

Figure 11-32 shows the layout of controls and textual content in `default.aspx`. The LoginView control displays its smart tag action menu items with the Edit Templates action selected. Clicking this link opens another smart tag action window with a drop-down box that lets you choose from the views available. As a minimum, you will have separate views for logged-in users versus anonymous users. If you implemented roles (this sample did not do that), you could have separate view templates for each defined role.

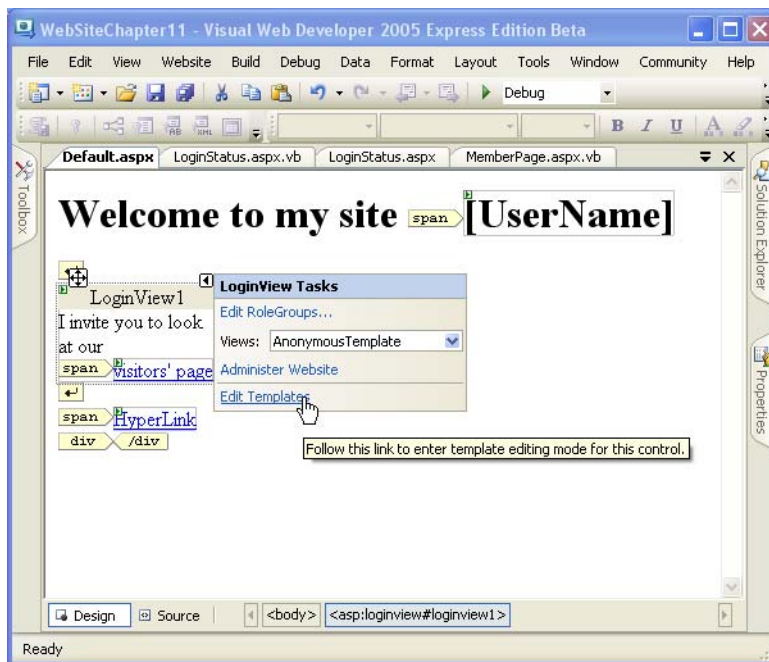


Figure 11-32. The LoginView control lets you edit templates that show different content to different groups of users.

Figure 11-33 shows the Anonymous Template open for editing with some content within it. If an anonymous user views `default.aspx` in a browser, the user will see the content in the Anonymous Template. If a logged-in user views `default.aspx` in a browser, the user sees the content in the LoggedIn Template.

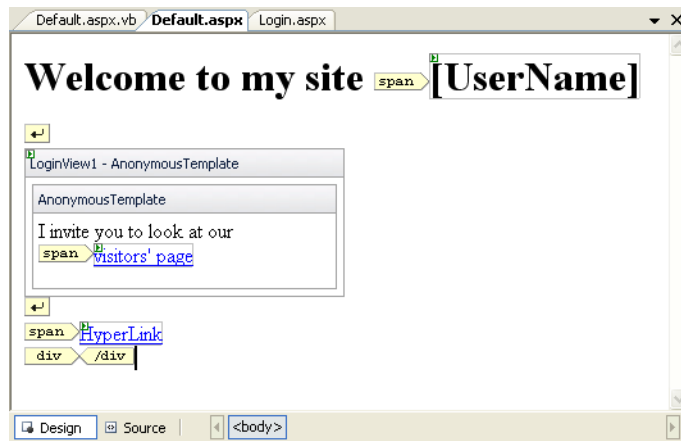


Figure 11-33. By editing LoginView templates you can designate content that shows conditionally based on the type of user.

Figure 11-34 shows the `default.aspx` page open by a user who logged in as Virginia Dobson. Notice from the Anonymous Template in Figure 11-33 that anonymous users see only a single link, which points at the `VisitorPage.aspx` file. Virginia Dobson is a logged-in user. Because the LoggedIn Template contains links to two pages, she has two links available to her. The members' page link is to the `MemberPage.aspx`. Users who connect anonymously to `default.aspx` do not see the link to `MemberPage.aspx`.

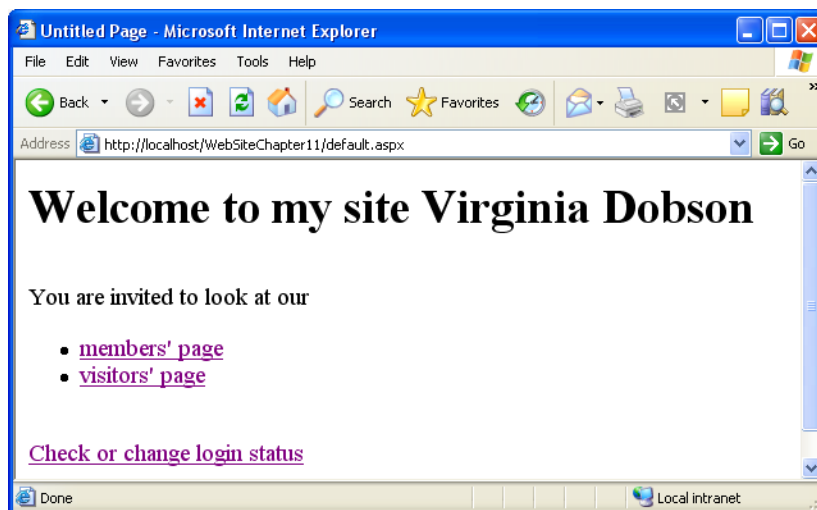


Figure 11-34. With Login controls, such as the LoginView control, and some optional code you can make pages dynamic depending on the login status of a Web site visitor.

The Hyperlink control at the bottom of the page is also dynamic. The Load event procedure for `default.aspx` contains code that differentially sets the `Text` and `NavigateURL` properties for the Hyperlink control based on the login status of the user.

Recall that the `Name` property of the `My.User` object reflects the value of the `LoginName` control. If the `Name` property of `My.User` is an empty string (""), then the current user is anonymous. Otherwise, the current user is a logged-in user, such as Rick Dobson, Virginia Dobson, or Tony Hill in our example. The following code excerpt sets the `Text` and `NavigateURL` properties for `Hyperlink4` based on the login status of the current user. The code segment assesses the login status of a user based on the `Name` property value of the `My.User` object.

```
If My.User.Name = "" Then
    Me.HyperLink4.Text = "Login for more access"
    Me.HyperLink4.NavigateUrl = _
        "http://localhost/WebSiteChapter11/login.aspx"
Else
    Me.HyperLink4.Text = "Check or change login status"
    Me.HyperLink4.NavigateUrl = _
        "http://localhost/WebSiteChapter11/loginstatus.aspx"
End If
```

Note You may wonder why the preceding code segment uses `My.User.Name` instead of just directly using the `LoginName` control. The answer is, the `LoginName` control does not function inside Visual Basic .NET code. The Login controls, including the `LoginName` control, are meant for visual display on a Web form page. As you'll discover, some Login controls have properties and events that you can process programmatically, but the `LoginName` control does not have a property for returning a logged-in user's name.

Modifying the Operation of a LoginStatus Control

The `LoginStatus.aspx` page demonstrates how to use and customize a `LoginStatus` control. The `LoginStatus` control detects a user's login status and displays a link; the appearance and functionality of the link change depending on whether the user is logged-in anonymously or with a user name.

- * When an anonymous user encounters a page with an uncustomized `LoginStatus` control, clicking the link takes the user to the Login page. The link's default text for anonymous users is login. After a user clicks the control, control transfers to the `Login.aspx` page (unless you modify the default application setting for a Login page).
- * When a logged-in user encounters a page with an uncustomized `LoginStatus` control, the control's link initially reads log out. A click to the link logs out a user. Then, the link's `Text` property changes to log in. A second click transfers control to the Login page so that a user can log in again with the same or a different user name.
- * You can change both the login and logout text for the `LoginStatus` link control from the Properties window or with code behind the page. Similarly, you can change the logout behavior with code behind the form. The `LoginStatus.aspx` page demonstrates one approach to achieving both of these goals.

Figure 11-35 shows the `LoginStatus.aspx` page. As you can see, the page has three controls: `Label`, `LoginName`, and `LoginStatus`. The `LoginStatus` control is selected so you see the left-facing arrow for displaying its smart tag actions. These merely let you see the default login and logout link text. The Properties window to the right of the Web form shows the default `LoginText` and `LogoutText` properties. The `Load` event for the page will modify both of these property settings based on the value of the `Name` property for the `My.User` object.

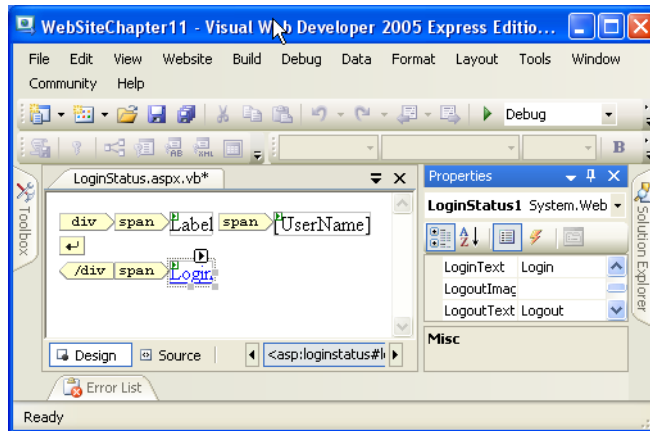


Figure 11-35. You can modify control properties at design time in the Properties window, but to make property settings dynamic at run time, you must alter them programmatically.

The following excerpt shows the code behind the `Load` event procedure for the `LoginStatus.aspx` page. This is an adaptation of the code for the `default.aspx` page. Two properties are set to one pair of values if the user is not an anonymous user and to another pair of values if the user is an anonymous user. One property is the `Text` property for the `Label` control. The second property is either the `LogoutText` property or the `LoginText` property depending on whether the user is logged in anonymously or with a user name.

```
If Not (My.User.Name = "") Then
    Me.Label1.Text = "You are logged in as: "
    Me.LoginStatus1.LogoutText = "Login as a new user."
Else
    Me.Label1.Text = ""
    Me.LoginStatus1.LoginText = "Click here to login."
End If
```

Note You may be wondering why we set the control properties in the preceding code excerpt for a user that's logged in anonymously. After all, users only see the link for the `LoginStatus` page on `default.aspx` if they are not anonymous users. However, if an anonymous user discovers the URL for the `LoginStatus.aspx` page by any means, the page should make sense or the user should be routed to another page. The next section shows how to route a user to another page.

What makes the `LoginStatus` page special is its use of the `LoggedOut` event for the `LoginStatus` control. The sample's `LoggedOut` event procedure transfers control directly to the

Login.aspx page instead of showing the LoginStatus control link with a text of log in, which is the default behavior for the control. When the LoginStatus.aspx page initially opens for a user with a user name, the control offers a link that allows the user to specify that she wants to log in as a new user. Therefore, there is no need to offer them a second link to confirm a user wants to log in. The LoggedOut event procedure requires just one instruction, which appears below, that transfers control to the Login.aspx page after the LoggedOut event fires.

```
My.Response.Redirect _  
    ("http://localhost/WebSiteChapter11/login.aspx")
```

Figure 11-36 shows the LoginStatus.aspx page after a user who logged in as Tony Hill transfers to the page from the last hyperlink on the default.aspx page. The text for the link on the LoginStatus.aspx page is for the LoginStatus control's LogoutText property. Clicking the link directs the browser session to perform two actions:

- * Log out the user
- * Transfer control to the Login page

It takes the code in the LoggedOut event procedure to make the second action happen without requiring a second click of a link.

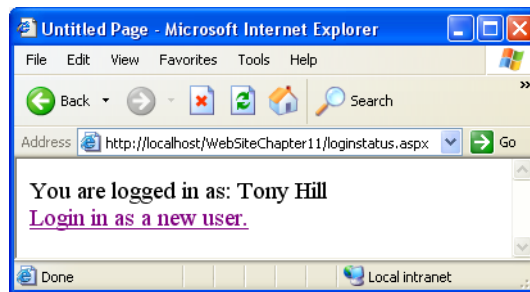


Figure 11-36. You can modify both the appearance and the behavior of a LoginStatus control with code behind a page.

Securing Page Content and Redirecting Access

It is common to want to have one or more pages at a site available exclusively to a select set of visitors, such as those who log in with a user name. The availability of the MemberPage.aspx and VisitorPage.aspx pages at the Web site helps to illustrate how to achieve this kind of result. One step to achieving the sought after outcome is to offer links to the special pages only to those authorized to view the page. However, users can sometimes find or guess the URL page even when you do not explicitly offer a link to the page. For example, an authorized user can share the link with someone who is not authorized to use the link.

The default.aspx already exposes the link to the MemberPage selectively with a LoginView control. Contrast Figure 11-33 with Figure 11-34 to see the different links exposed to anonymous versus logged in users. In addition, the Load event procedure associated with the MemberPage.aspx page can redirect attempts to view it to the VisitorPage.aspx page. It takes just a single instruction (My.Response.Redirect) nested inside an If...Then statement to route anonymous users to the VisitorPage.aspx file when they

attempt to navigate to the `MemberPage.aspx` file. The following excerpt shows the code inside the `Load` event procedure in the `MemberPage.aspx.vb` file.

```
If My.User.Name = "" Then
    My.Response.Redirect _
        ("http://localhost/WebSiteChapter11/VisitorPage.aspx")
End If
```

Change and Recover Passwords with Restricted-Access Folders

The samples for this section focus on two topics: managing access to folders at a Web site for logged in versus anonymous users and managing passwords for logged-in users. These two topics relate to one another because you can secure password management by locating at least one type of page (the one for changing passwords) in a folder with access restricted to logged-in users. On the other hand, a page that initiates sending a password to enable for a user who forgets her password to log in needs to be in a folder available to anonymous users. If a user forgets her password, the only way that she can connect to a Web site is as an anonymous user.

The samples in this section use a different `default.aspx` file than any of the preceding `default.aspx` files. To set up the samples in this section, copy the contents of `default3.txt` in the `WebSiteChapter11` Web site to the `default.aspx` file for the Web site you are using to run the samples in this section. Then, remove the `Load` event procedure code from `default.aspx.vb`.

Restricting Folder Access

The technique described in the “Securing Page Content and Redirecting Access” section is adequate for securing access to one or a few Web form page files. However, if you have many files that require restricted access, copying the code into the `Load` event procedure for each file can quickly become cumbersome. Furthermore, by taking advantage of Visual Basic .NET coding conventions (`My.User.Name`), the technique is applicable only to selected types of files, such as Web form pages, and not other types of files that a Web folder can have, such as `.gif` files or even `.htm` files.

To simulate and demonstrate the scenario of multiple folders, create two new regular folders – one named `Members` and another named `Visitors`. You can create a regular folder by right-clicking a Web site’s root folder in Solution Explorer and choosing `Add Folder ~TRA Regular Folder`. VWDE assigns a tentative name of `NewFolder1` (or some other number, such as `NewFolder2` or `NewFolder3`, etc). Replace the tentative name for a regular folder by typing a more meaningful name, such as `Members` or `Visitors`.

After creating the `Members` and `Visitors` folders, you can restrict access to the `Members` folder so that only non-anonymous users can access it. In other words, content in the `Members` folder is for members only. You can restrict access to a whole folder of files with the ASP.NET Web Site Administration Tool. Recall that you can open this tool by choosing `Website ~TRA ASP.NET Configuration` from the VWDE Standard menu. After opening the tool, click the `Security` tab and then click the `Create Access rules` link in the `Access rules` panel. These clicks open a Web page with a form having a heading of `Add New Access Rule`.

The `Add New Access Rule` form has three parts from left to right. The left part lets you specify to which folder or directory an access rule applies. The middle part enables you to

designate to which group of users the rule applies. The right part specifies one of two access permissions – Allow or Deny.

By default, all users, even anonymous ones, have access to all folders. Therefore, to restrict access so that only logged-in users can access the **Members** folder, deny access to the **Members** folder by anonymous users. Follow these instructions depicted in Figure 11-37 for implementing the rule from the Add New Access Rule form.

- * Select the **Members** directory on the left side of the form.
- * Select the Anonymous users from the middle part of the form.
- * Select Deny in the right part of the form.

After making your selections, commit your new access rule by clicking OK. Then, exit the ASP.NET Web Site Administration Tool by closing its browser session. Once you commit the access rule, only logged in users will be able to access the Web pages and other content in the **Members** folders.

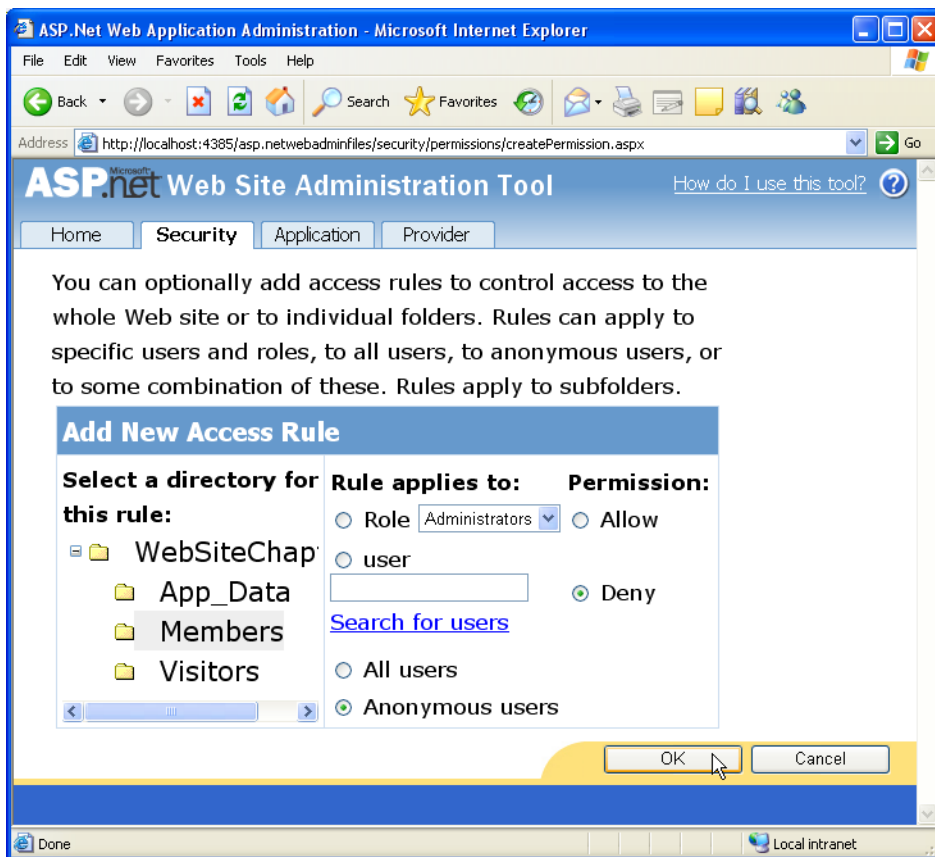


Figure 11-37. You can create custom access rules for Web site folders with the ASP.NET Web Site Administration Tool.

Changing Passwords

Whenever you implement a system with passwords, one common requirement is to allow users to change their password. In fact, the approach for recovering passwords with Login controls makes using the ChangePassword control highly desirable. You can drag the ChangePassword control from the Login group of the Toolbox to a Web form page.

The ChangePassword control includes three TextBox controls and a couple of Button controls. In order to change your password with the ChangePassword control, you must perform three steps.

- * Input your current password. This essentially confirms you have authority to change the password.
- * Designate a new password and confirm it.
- * Click a button to commit your change.

A Web form page to change a password is a good example of the type of capability that should be restricted to logged-in users. After all, anonymous users do not even have a password because they didn't log in. Therefore, you can create a new Web form page named `ChangePassword.aspx` in the `Members` folder. If you applied the access rule defined in the "Restricting Folder Access" section anonymous users will not be able to access the `ChangePassword.aspx` file. In fact, if an anonymous user does try to access the page, they are automatically transferred to the `Login.aspx` page.

Figure 11-38 shows the `ChangePassword.aspx` page on its Design tab. Notice the tab for the page starts with `Members/`. This prefix indicates the page resides in the `Members` folder of the Web site. The heading at the top of the page asks the user if they are sure they want to change their password. The use of a `LoginName` control helps to personalize the question. The view of the ChangePassword control on the Design tab shows an error message about matching passwords that appears if the new and confirming password do not match. Since the password characters do not display on the control, this requirement for a confirming password helps to guard against a user locking herself out of an account as she attempts to change her password.



Figure 11-38. This Design tab view of the `ChangePassword` control bundles together `TextBox` and `Button` controls along with an error message.

Figure 11-39 displays a completed `ChangePassword.aspx` page. All three text boxes hide their entries because they contain passwords. Three conditions must be met for the attempt to change a password to succeed.

- * The contents of the Password box must match the existing password at the time you enter page.
- * The contents of the New Password and Confirm New Password must match one another.
- * The contents of the New Password box must be a strong password.

If any of the preceding conditions are not satisfied, an error message appears. At that point, a user can re-enter characters into one or more boxes. If all conditions are valid, the user receives a message confirming success. Clicking the Continue button on the confirmation page, transfers control to `default.aspx`. The next time that a user logs in they must use the new password.

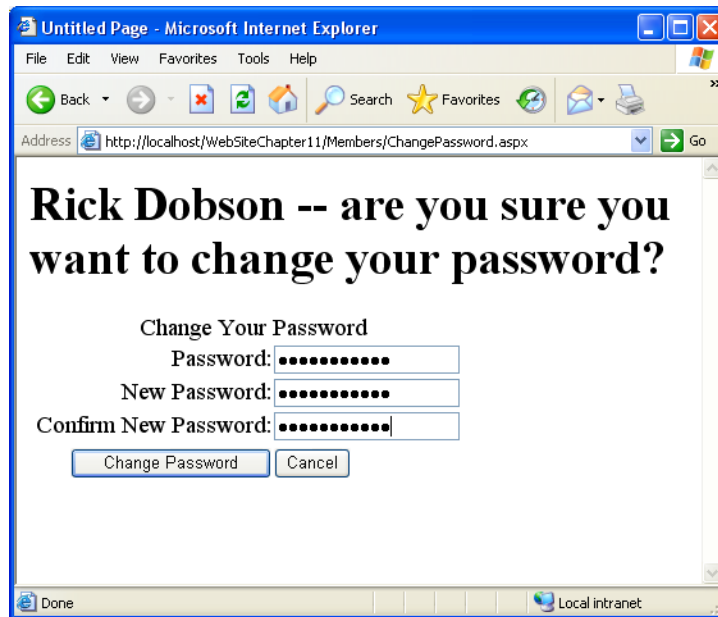


Figure 11-39. The `ChangePassword` control has builtin functionality to verify the input to it as well as to change the password for a user if its input is valid.

Recovering Passwords

A `RecoverPassword` control helps users who forgot their password get a new password so that they can log in to the Web site again. In order for a user to get a new password emailed to his email address, he must know his user name and the answer to his security question. The Create User form in Figure 11-25 shows the inputs for a user named Rick Dobson along with an email address, a security question, and its answer. This form is from the Security Setup Wizard, but you can access the same Create User form with the Create user link on the Security Tab of the ASP.NET Web Site Administration Tool.

Several conditions need to be in effect for the RecoverPassword control to work. These include the following.

- * The Web site needs to be configured to use a SMTP server for sending email. The user receives a new password at the email address specified in the Create User form.
- * The Web site must allow anonymous access so that a user who forgot his password can get to the form with the RecoverPassword control at the Web site.
- * The user must have a security question associated with his user name, and he must know the answer to it.

The RecoverPassword control has three different displays associated with it. Initially, the control prompts for a user name. After a reply with a valid user name, the control prompts the user with the security question for the user name. If the user inputs a valid answer to the security question, the control shows a success screen and mails a new password to the email address associated with the user name.

Note You may be wondering why the RecoverPassword control does not just send the original password that the user forgot. The answer is that VWDE does not save the original password, but a hashed version of it. There is no way to recover the original password from its hash. Because hash codes are unique for a given input, VWDE can compare the hash code for a password sent in a browser to the internal hash code to verify if a password is valid.

When the user receives the new mailed password, he can log in with the new password and his user name. The new password is a strong password that is likely to be difficult to remember, but it is adequate for one-time use. Later, the user can navigate to a Web form page with the ChangePassword control and create a new password that is both strong and easier to remember.

You will want to save the page containing the RecoverPassword control in a location at a Web site that is accessible to anonymous users, such as in the **Visitors** folder created in the “Restricting Folder Access” section. If you fail to store the page in a location accessible to anonymous users, a user who forgot his password will not be able to access the page for recovering his password. The sample application for the RecoverPassword control has the control on the **RecoverPassword.aspx** page in the **Visitors** folders. Figure 11-40 shows the first of the three displays associated with the RecoverPassword control.

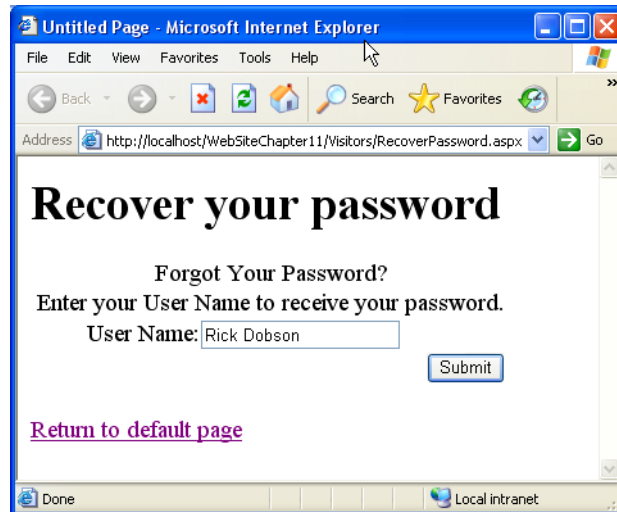


Figure 11-40. The RecoverPassword control automates sending a new password to a user who forgot his current password.

Figure 11-41 shows the message mailed by VWDE to the email address for the user named Rick Dobson. The user name pertains to the user's name at the Web site not to the recipient's name. Notice how strange the characters are in the new password (o%&PKcZJyde/]U). This kind of strong password is what recommends the best practice of making available a capability for users to change their password. In this way, a user who forgot his password can use the new password to log in, and then immediately change the password to one that is both strong and a lot easier to remember.

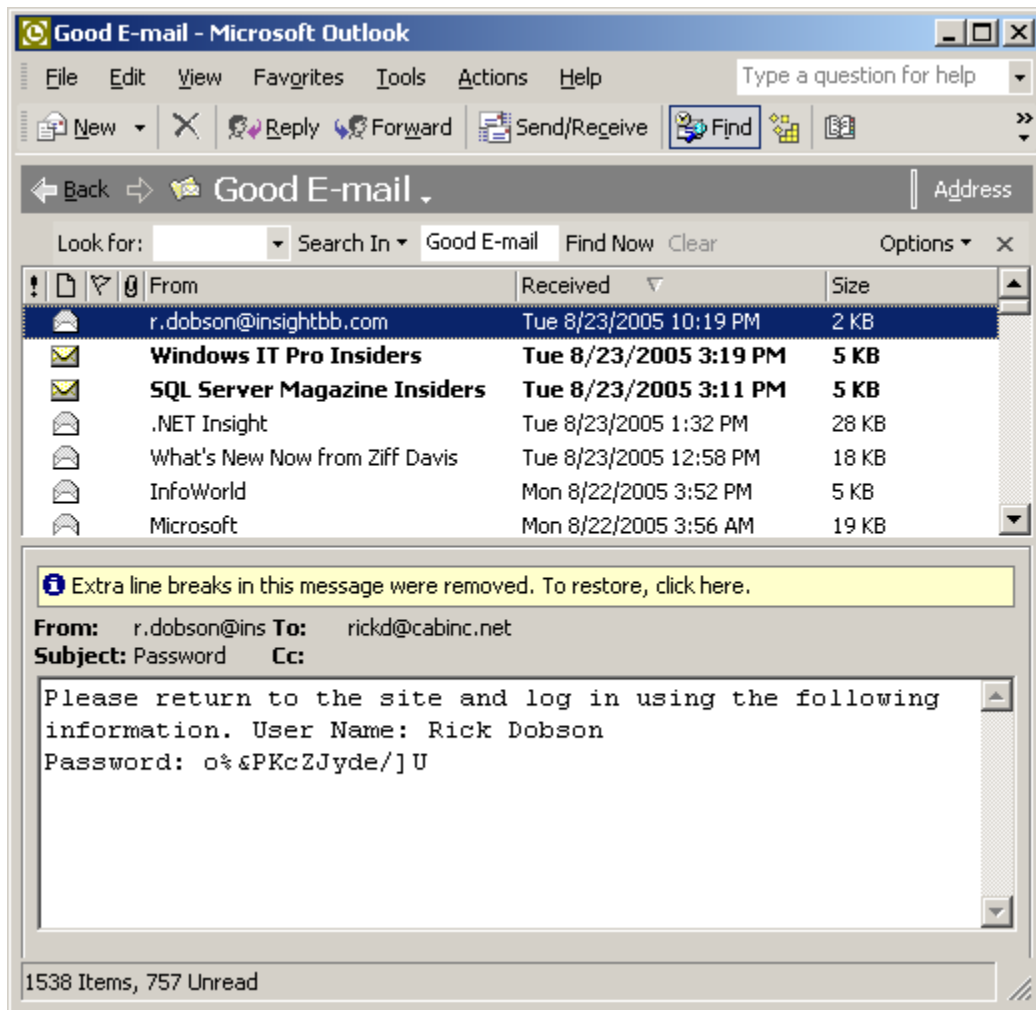


Figure 11-41. An email message in Microsoft Outlook with the email message sent from a RecoverPassword control.

Summary

This chapter introduced you to three special ASP.NET 2.0 development features:

- * User controls
- * Master pages
- * Login controls

The coverage of user controls focused on the creation and re-use of three user controls. In examining the steps for creating and re-using these three controls, you saw how to put user controls to work in your own solutions.

Master pages help you to format the pages throughout your Web site. This chapter's coverage of Master pages illustrated how to add formatting to a Master page, including text, graphics, and controls, and then make that formatting available through other pages that are

based on the Master pages. The coverage of the Menu control in the section on the Master pages deserves special mention because it illustrated a simple solution to a common problem.

The focus on Login controls and the ASP.NET Web Site Administration Tool demonstrated how simple it is to create users and customize Web site content based on user identity. You also learn how to manage passwords by enabling users to change their own password or even recover a password that they forgot.