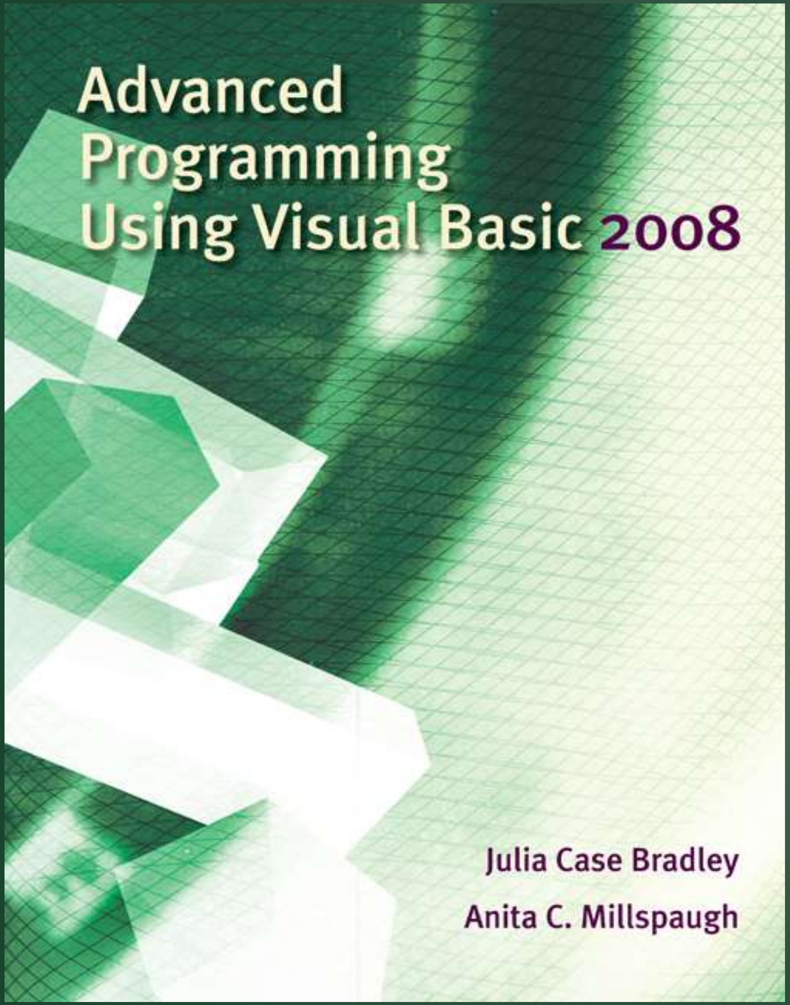


Chapter 11

User Controls

Objectives

- Create a Windows user control using inheritance from an existing control type
- Add a new user control to a form
- Add properties to a user control
- Raise an event in a control class and write code to handle the event in a form
- Create a new composite control by combining preexisting controls
- Create a Web user control and add it to a Web page

Windows User Controls

- User controls are created by the programmer
- A *composite control* is made up of more than one individual control
- *Constituent controls* are the individual controls that make up the composite control
- New user control can be added to the toolbox and used in other Windows projects

The Control Author versus the Developer

- The *author* creates, tests, and compiles the control and it appears in the toolbox
- The *developer* uses the control
- The author of a control must plan for the design-time *and* run-time behavior of the control

Creating a New Control - 1

- Begin a new project based on the Windows Forms Control Library template
  - – Not available in Visual Basic Express Edition
- Controls created in this type of project may be used in multiple Windows projects
  - – Add a new UserControl to an existing project if the control will only be used in the current project
- The new UserControl object appears as a design surface, similar to a form in a Designer window

Creating a New Control - 2

- Design the visual interface in the Designer window
  - Drag constituent controls to the design surface
  - The visual representation of the control will not be seen for an inherited control
- View and modify the created class for the control in the Code Editor window
  - Class automatically inherits from the UserControl class
    - See the UserControl.Designer.vb file

Inheriting from an Existing Control - 1

- Inherit from most Windows Forms controls except the Progress Bar
- New controls inherit all properties, methods, and events of the base class
  - Code can be written to override the behaviors
  - Add new properties, methods, and events for the derived control

Inheriting from an Existing Control - 2

- Create a customized control
  - – Create a project based on the Windows Forms Control Library template
  - – Modify the class name and the *Inherits* clause in the Designer.vb file
  - – Add additional desired functionality
  - – Build the DLL
    - After the DLL is created, create a Windows project to test the new control

Creating an Inherited User Control – Step-by-Step

- Create a new project
- Add an event handler
- Build the project
- Test the user control in a form
- Add controls to the form
- Run the project

Adding Properties to a Control

- Set up new properties of a control class just like other classes
- Declare a module-level private variable to hold the property and create Property procedures
  - Properties appear in the Properties window when an instance of the control is added to a form

Setting Default Values for Properties

- Set a property variable to an initial value
  - Provides the property with a default value
- Add an instance of the control to a form
  - The default value appears in the Properties window
  - If the developer changes the value of the property, the new value is retained in the property
  - The control is initialized only once when added to the form
  - Changes made at design time are retained
- The value of a ReadOnly property cannot be changed
  - The code for the control can change the value

Adding Events to a Control - 1

- Most objects can generate events
  - Also called *raising events* or *firing events* such as Click, DoubleClick, MouseUp, and Move
  - Events are caused by user action
    - A click or mouse move
  - Other events are generated by the system
    - A timer firing or the Form_Load event
- Objects must either raise an event or throw an exception to which the form module can respond

Adding Events to a Control - 2

- An object that generates or raises an event is called the *event source* or the *event provider*
- The object that responds to an event is called an *event sink* or an *event consumer*
  - – User clicks a command button and form's Button_Click event handler executes
    - Command button is event source
    - Form is the event sink

Raising Events

- Declare the event in the Declaration section of the class, include any arguments to be passed

```
Public Event InvalidDate(ByVal Message as String)
```

- Raise the event in code
  - When a condition triggers the event, use the *RaiseEvent* statement

```
If Not DateTime.TryParse(Me.Text, TestDate) Then
        ` Invalid date format, raise an event.
        RaiseEvent InvalidDate("Invalid date.")
End If
```

Responding to Events

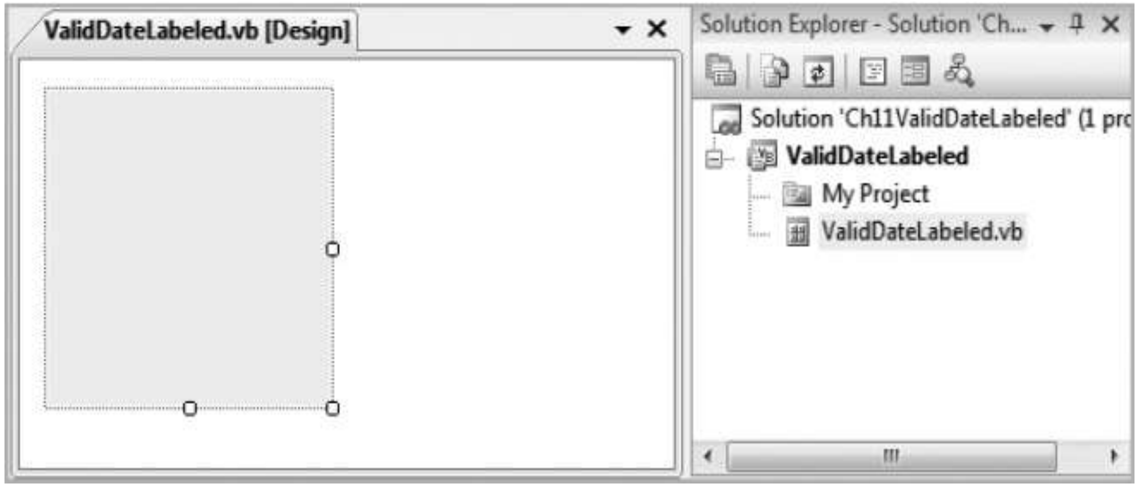
- Any class can be an event sink and respond to events raised by an event source
- Add a user control to a form
  - Drop down the Methods list in the Editor window
  - The event appears on the list
- Write the code that is to execute when the event fires

Creating a Composite User Control

- Used to combine multiple controls into a single user control
  - Combine the ValidDateTextBox control with a label
    - Label can have a default Text property that can be modified by the application developer

Create a New Composite User Control

- Begin a new project based on the Windows Forms Control Library template
  - Leave the inheritance as a UserControl
  - Use the control's visual designer, treat it just like a form
  - Add as many controls as needed
  - Name the constituent controls and refer to them as in any application


ValidDateLabeled.vb [Design]
Solution Explorer - Solution 'Ch...
Solution 'Ch11ValidDateLabeled' (1 pro
ValidDateLabeled
My Project
ValidDateLabeled.vb

Adding a Control to the Toolbox

- Controls that have already been developed are not automatically added to the toolbox
  - Right-click on the toolbox and select *Choose Item*
  - In the *Choose Toolbox Items* dialog box, on the *.NET Framework Components* tab, click on the *Browse* button
  - Browse to find the control's .dll file in its bin\Debug folder
  - Select the .dll file
  - Close the dialog box and the control appears in the toolbox

Adding Constituent Controls

- Add any controls or components from the toolbox to the design surface of the composite control
  - Set the constituent controls to anchor to all four edges of the user control
    - Interior controls will resize when the user control is resized

Exposing Properties of Constituent Controls

- Properties of the constituent controls are available inside the composite control, but not to the application developer
- The control author determines which properties to expose to the developer

Exposing the Events of the Constituent Controls

- Any events of the constituent controls are available in the code of the composite control
  - Events are not available to the form on which the control is placed
  - Must declare the event in the composite control and pass the event along

```
' Declare the event at the module level.
Public Event InvalidDate(ByVal Message As String)
```

- Code is written in the event handler for the constituent control
  - Raise the event or write additional code

Using the Composite Control

- After the control is created, test it in a form in the same manner as the inherited control
- Add a new project for the test form
  - Add a reference to the project that holds the composite control
  - Add the control to the form
  - Use the *Choose Items* command if the new control does not appear in the toolbox
- If modifying the control, close the user control's designer before rebuilding
- Rebuild the solution and re-add the control to the form to get the updated control

Web User Controls

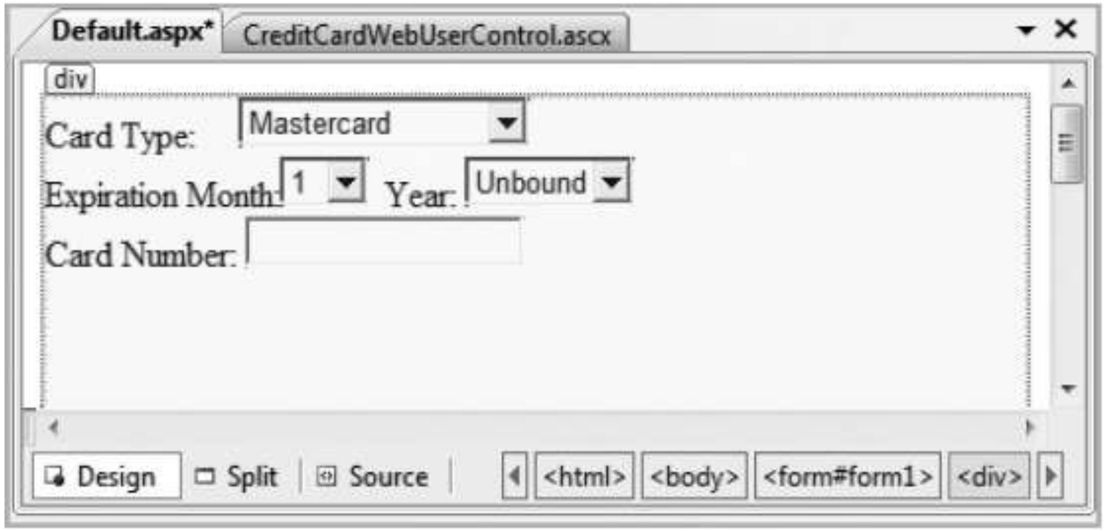
- Web user controls work differently than Windows user controls
  - Think of a Web user control as a “mini-page” that can be displayed on many other pages
- Create reusable pieces of an interface that contain HTML controls, server controls, and code
- Use the ASP.NET Web Site template and add a WebUserControl to the project

Creating a Web User Control – Step-by-Step - 1

- Create the project
- Design the user interface
- Set up properties to allow the Web page to retrieve the values a user enters
  - Properties can be read only unless the Web page must be able to set initial values
  - When the value from a control holds the property, it is not necessary to declare module-level variables to hold the property values

Creating a Web User Control – Step-by-Step - 2

- Compile the control
- Test the control
  - Drag the user control file from the Solution Explorer to the form's design surface
  - Run the project



Card Type: Mastercard
Expiration Month: 1 Year: Unbound
Card Number: