

## .NET Custom Controls

## Custom Controls

- Lots of controls are built into .NET, but you may want to design your own control
- Three ways of doing it:
  - Customize an existing control
    - Derive from an existing control and override its methods, add your own properties
      - E.g., a button that counts how many times it's been clicked
  - Combine functionality of several existing controls by deriving from UserControl
    - E.g. combine text and label controls to create an address control that captures user input
  - Create a brand new control by deriving from base class Control
    - Most flexible, but most complex way of doing it
    - You're responsible for drawing it
    - And overriding Control base class event handler functions

## Customizing an Existing Control

- A customized button control that counts how many times it's been clicked

– File | New | Project | Visual C# | Windows Control Library

- Name: CountedControl
- Right Click to view the code
  - Change partial class name to CountedButton and base class to Button
  - Change filename from UserControl1.cs to CountedButton.cs
  - In CountedButton.cs change partial class name to CountedButton
  - Add a private class level variable numClicks initialized to 0
  - Add an override the OnPaint(PaintEventArgs e) method

```
protected override void OnPaint(PaintEventArgs e)
{
    if (numClicks == 0)
        this.Text = "Never been clicked";
    else
        this.Text = "Clicked " + numClicks + " times";
    base.OnPaint(e);
}
```

- Add an override OnClick() Handler to update numClicks and invalidate the button so it will be repainted

```
protected override void OnClick(EventArgs e)
{
    numClicks++;
    Invalidate();
    base.OnClick(e);
}
```

- Add a NumClicks property to the control

```
public int NumClicks
{
    get { return numClicks; }
    set { numClicks = value; }
}
```

- Build the control  $\neq$  CountedControl.dll

## Using the New Control

- Two ways of incorporating a new control into a Windows Form from Visual Studio:
  1. Add a reference to the control:
    - In Windows Explorer right click on "References" and choose "Add Reference"
    - Click on "Projects" tab and click "Browse" button
    - Navigate to the dll, double click on it, and click "OK"
  2. Add the control to the tool box:
    - "Tools" | "Choose Toolbox Items"  $\neq$  "Choose Toolbox Items" dialog box
    - Choose the ".NET Framework Components" tab and the "Browse" button
      - Navigate to the dll and double click on it
    - It will appear at the bottom of your toolbox
    - You can then drag it over and use it like any other tool in the toolbox
- Example: CountedControlTester
  - Has a CountedControl Button and a label that displays the NumClicks property value every time new button is clicked

## Creating a UserControl

- Combine functionality of several existing controls by deriving from class UserControl
- Example: A "clock" user control
  - Consists of a timer and a label to display the time
- Create a new Form to display our custom control
- Create a UserControl class for the project
  - Project | Add User control
    - Type in ClockUserControl and click "Add"
  - Drag needed controls over to the resulting "form"
    - A timer and a label
    - Set timer's Enabled property to true and Interval to 100
    - Add a timer tick event handler
  - Note that the ClockUserControl is in the tool box
  - Drag it over to the main Form just like any other control
- Could also have done this as a Windows Control Library Project as in the previous example

## Creating a New Control from Scratch

- Derive the control from base class Control
  - This is the base class for all controls
    - It defines no specific behavior
    - Programmer must draw the control
      - And determine its exact look, feel and behavior
      - Must program every aspect of the control
  - Control class does do basic event handling involved in all controls
    - Usually event handlers for interesting events like the Paint event should call the base class OnPaint() handler
    - Then add code to for custom graphics
- This approach give the most flexibility, but requires the most programming

## Example of a New Control

- DisplayNameControl: Spells out a person's name
  - Create a new Windows Control Library project
  - Resize it & change any properties now
  - Change class name and derive it from Control
    - Also change file name
      - Note that Form Design Window goes away – you must hand code
    - Comment out 2 lines w/ AutoScale... at top of InitializeComponent()
  - Add a public string to hold the name to be displayed
  - In the class constructor give the string an initial value
  - Add a public method to extract and draw each character in the string
    - Use the string object's Substring(start,count) method to extract each single-character substring
  - Override the base class OnPaint() method so it draws the current string
  - Build the project and use the "Choose Toolbox Items" dialog box to add the new control to the Toolbox

## Using the New Control

- DisplayNameTester example
  - A Windows Form application
  - Whenever user clicks a button, whatever name is in a textbox is spelled out in a DisplayName control
  - Drag over a DisplayName control, a label, a textbox, and a label from the tool box
  - Add a button click handler that extracts the text from the textbox and uses the DisplayName control's DrawName() member function to spell it out