Custom Controls and Forms

Contents

Varieties of Custom Controls

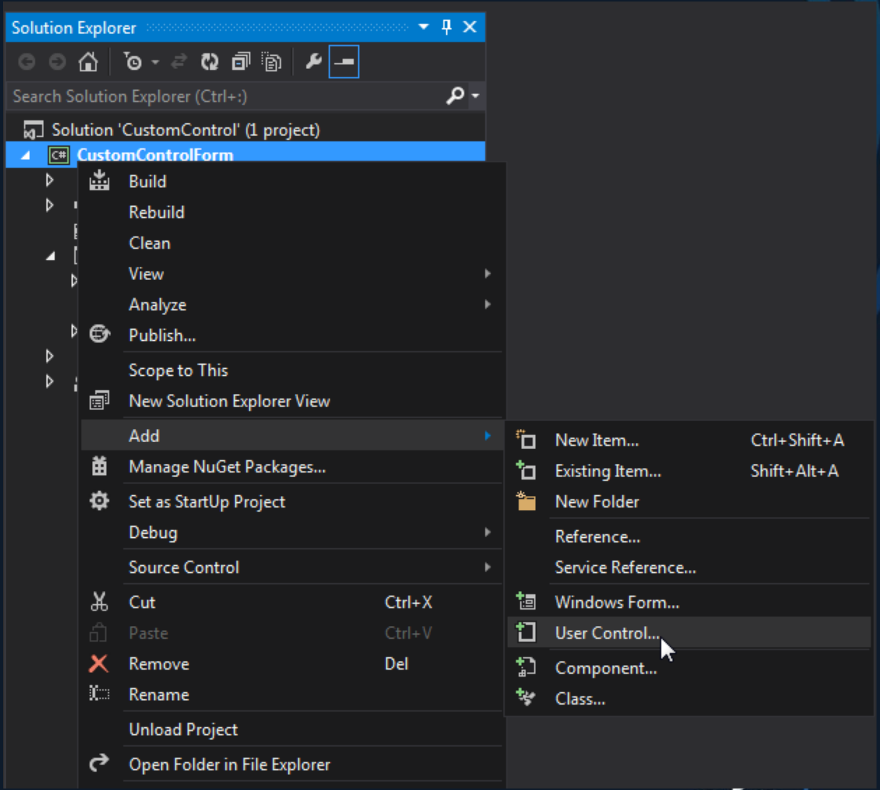
- Composite controls
  - A collection of controls contained in a common container
- Extended controls
  - Inherit from any existing control to extend functionality
- Custom controls
  - Created from scratch, specify your own painting

Composite Controls

- Inherits from the UserControl class
  - Provides keyboard routing for child components
  - Enables child controls to work as a group
  - Ensures child controls can receive focus
- Easiest custom control to create
- Package and reuse controls between applications

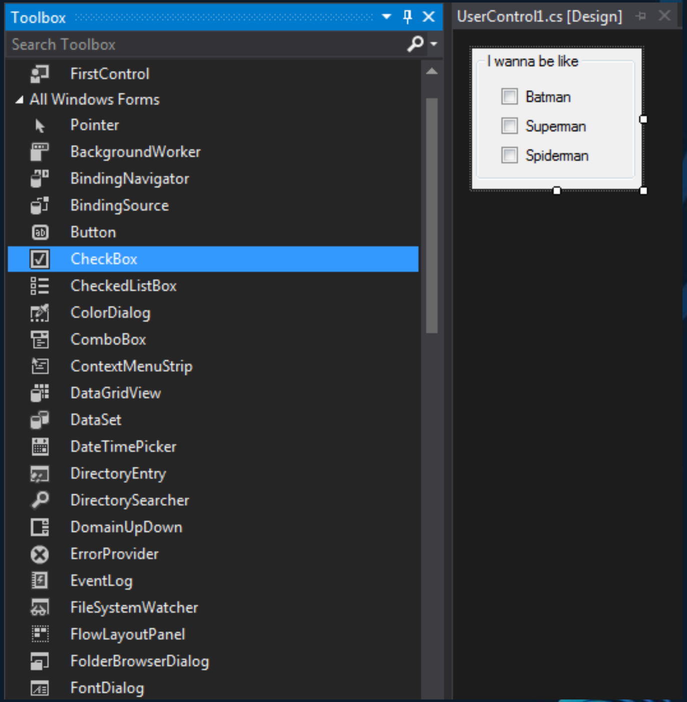
Composite Controls

- Create a Windows Forms project
- On the *Project* menu, select *Add User Control*
  - Or select from the Project context menu

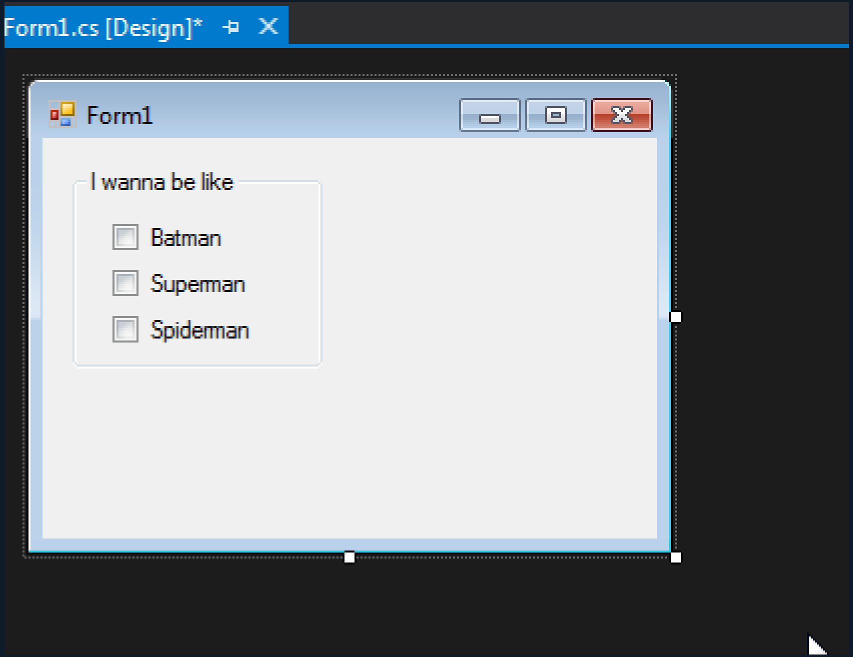


Composite Controls

- Add controls from the *Toolbox* to the composite control
- You must build the project for your control to appear in the Toolbox
- Add the composite control to the Form



Composite Controls



I wanna be like

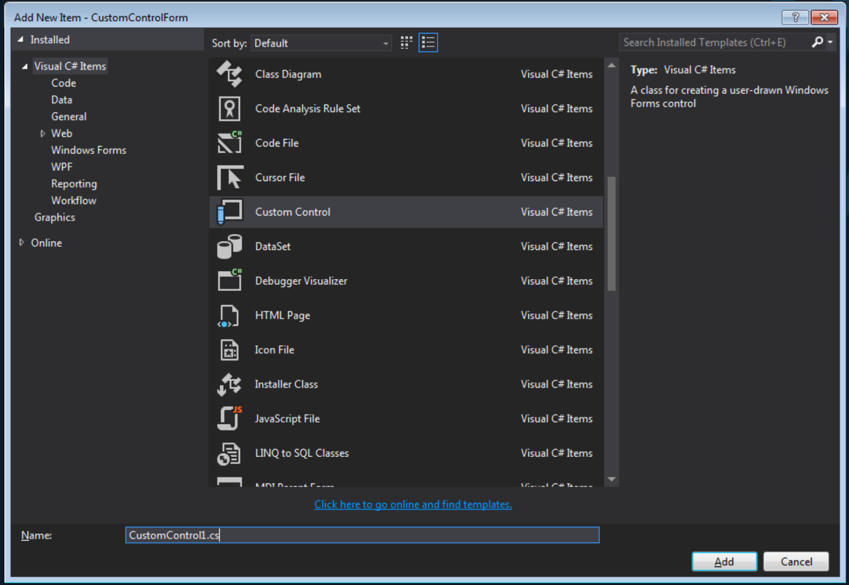
- Batman
- Superman
- Spiderman

Extended Controls

- Inherit from an existing control
- Retain and extend functionality
- Can override paint logic for custom appearance
- Use this method when:
  - The needed functionality is mostly identical to an existing control,
  - Don’t need a custom GUI, or
  - Want a new GUI for an existing control

Extended Controls

- Create a Windows Forms project
- On the *Project* menu, select *Add New Item*
  - Or select from the Project context menu
- In the *Add New Item* dialog, select *Custom Control*


Add New Item - CustomControlForm

Installed
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Workflow
  - Graphics
- Online

Sort by: Default

Search Installed Templates (Ctrl+E)

Class Diagram	Visual C# Items
Code Analysis Rule Set	Visual C# Items
Code File	Visual C# Items
Cursor File	Visual C# Items
Custom Control	Visual C# Items
DataSet	Visual C# Items
Debugger Visualizer	Visual C# Items
HTML Page	Visual C# Items
Icon File	Visual C# Items
Installer Class	Visual C# Items
JavaScript File	Visual C# Items
LINQ to SQL Classes	Visual C# Items

Type: Visual C# Items

A class for creating a user-drawn Windows Forms control

Click here to go online and find templates.

Name: CustomControl1.cs

Add Cancel

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

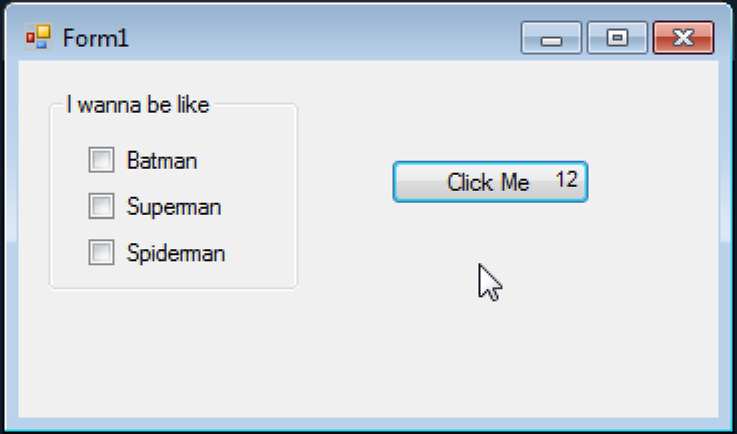
Extended Controls

- Open CustomControl1.cs in the Code editor
- Locate the class declaration
- Change the base class to the control you want to inherit from
  - Eg. Button
- Implement custom methods, properties, drawing

Extended Controls

```
public partial class CustomControl1 : System.Windows.Forms.Button
{
    int clickCount = 0;

    public CustomControl1()
    {
        InitializeComponent();
    }

    protected override void OnClick(EventArgs e)
    {
        base.OnClick(e);
        clickCount++;
    }

    protected override void OnPaint(PaintEventArgs pe)
    {
        base.OnPaint(pe);

        Font drawFont = new Font("Arial", 8);
        SolidBrush drawBrush = new SolidBrush(Color.Black);
        pe.Graphics.DrawString(clickCount.ToString(), drawFont, drawBrush, Size.Width - 20, Size.Height - 20);
    }
}
```

Extended Controls

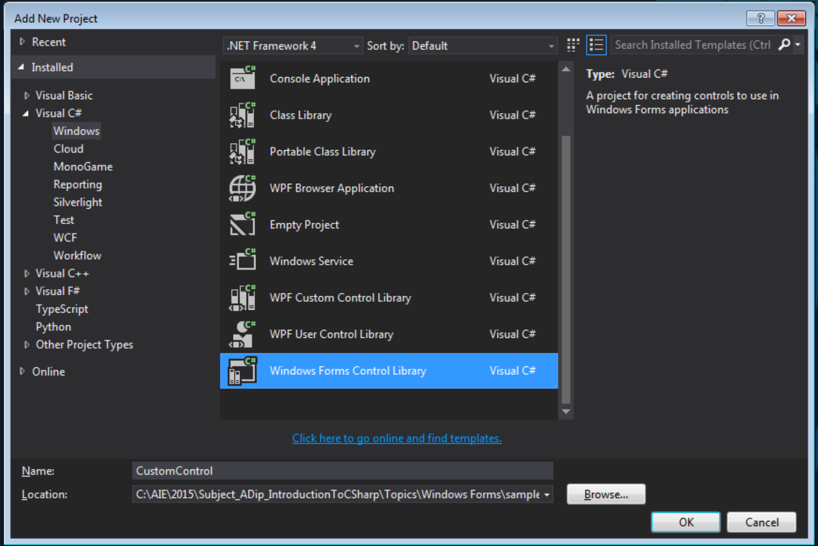


Custom Controls

- Created essentially from scratch
- Inherit from the *Control* class
- Greater flexibility, can tailor control to exact needs
- Must write code for the OnPaint event
- Use a Custom Control if:
  - Want to provide a custom graphical representation
  - Implement custom functionality not available in standard controls

Custom Controls

- Add a *New Project* to your solution
- Create a *Windows Forms Control Library*
- Remove the *User Control* and add a *New Class* derived from *Control*



Add New Project

- Recent
- Installed
  - Visual Basic
  - Visual C#
    - Windows
    - Cloud
    - MonoGame
    - Reporting
    - Silverlight
    - Test
    - WCF
    - Workflow
  - Visual C++
  - Visual F#
  - TypeScript
  - Python
  - Other Project Types
- Online

.NET Framework 4 | Sort by: Default | Search Installed Templates (Ctrl

Console Application	Visual C#
Class Library	Visual C#
Portable Class Library	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#
WPF User Control Library	Visual C#
Windows Forms Control Library	Visual C#

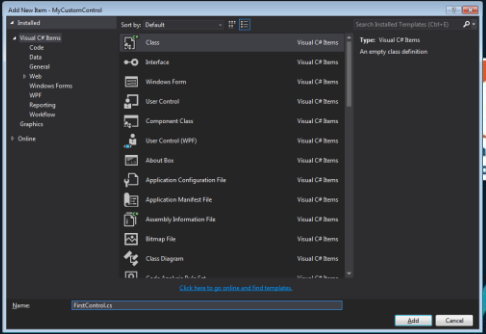
Type: Visual C#

A project for creating controls to use in Windows Forms applications

Click here to go online and find templates.

Name: CustomControl

Location: C:\AIE\2015\Subject_ADip_IntroductionToCSharp\Topics\Windows Forms\sample



Add New Item - MyCustomControl

- Installed
  - Visual C# Items
    - Code
    - Data
    - General
    - Web
    - Windows Forms
    - WPF
    - Reporting
    - Workflow
  - Graphics
- Online

Sort by: Default | Search Installed Templates (Ctrl+E)

Class	Visual C# Items
Interface	Visual C# Items
Windows Form	Visual C# Items
User Control	Visual C# Items
Component Class	Visual C# Items
User Control (WPF)	Visual C# Items
About Box	Visual C# Items
Application Configuration File	Visual C# Items
Application Manifest File	Visual C# Items
Assembly Information File	Visual C# Items
Bitmap File	Visual C# Items
Class Diagram	Visual C# Items

Type: Visual C# Items

An empty class definition

Click here to go online and find templates.

Name: FirstControl.cs

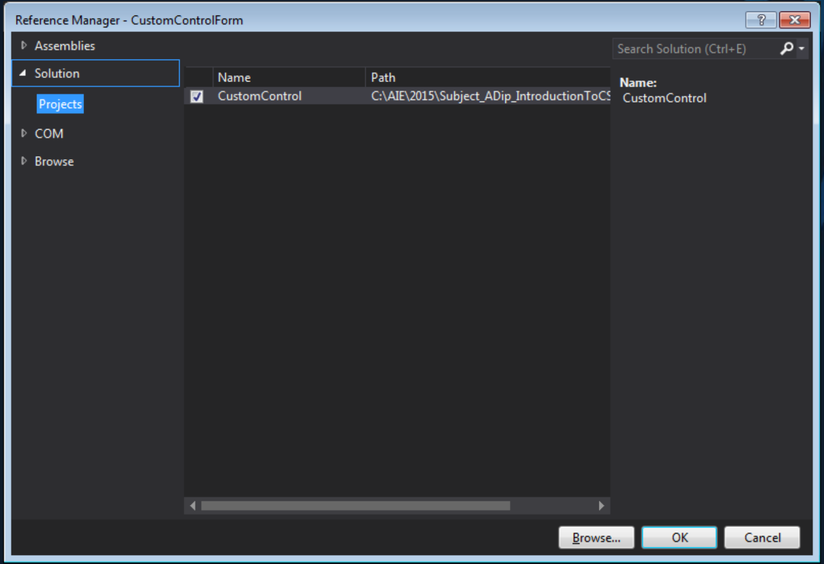
Custom Controls

- Add an image to the project
  - This is the background for the Analogue clock
  - Set properties to ‘Embedded Resource’, and ‘Do not copy’
- Code your custom control

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Windows.Forms;
using System.ComponentModel;

namespace CustomControl {
    public class FirstControl:System.Windows.Forms.Control {
        private Bitmap bitmap;

        public FirstControl() {
            DoubleBuffered = true;

            ClockTimer.Tick += ClockTimer_Tick;
            ClockTimer.Enabled = true;
            ClockTimer.Interval = 1;
            ClockTimer.Start();

            bitmap = new Bitmap(
                GetType().Module.Assembly.GetManifestResourceStream(
                "CustomControl.jellyfish_trans.png"));
        }

        private void ClockTimer_Tick(object sender, EventArgs e) {
            Refresh();
        }

        private Timer ClockTimer = new Timer();

        private bool showSeconds = true;

        [
        Category("Show Seconds"),
        Description("Show the second hand.")
        ]
        public bool  ShowSeconds {
            get {
                return showSeconds;
            }
            set {
                showSeconds = value;
                Invalidate();
            }
        }

        protected override void OnPaint(PaintEventArgs e) {
            base.OnPaint(e);
            e.Graphics.DrawImage(bitmap, 0, 0, this.Width, this.Height);

            float radius = (Size.Width / 2);
            PointF origin = new PointF(
                                     Size.Width / 2, Size.Height / 2);

            if(showSeconds == true)
                e.Graphics.DrawLine(Pens.Black, origin,
                                     PointOnCircle(radius,
                                     DateTime.Now.Second * 6f, origin));

           e.Graphics.DrawLine(Pens.Black, origin,
                                     PointOnCircle(radius * 0.75f,
                                     DateTime.Now.Minute * 6f, origin));

            e.Graphics.DrawLine(Pens.Black, origin,
                                     PointOnCircle(radius * 0.50f,
                                     DateTime.Now.Hour * 30f, origin));
        }

        private PointF PointOnCircle(float radius, float angleInDegrees,
                PointF origin)
        {
            float x = (float)(radius * Math.Cos((angleInDegrees - 90f) *
                Math.PI / 180F)) + origin.X;
            float y = (float)(radius * Math.Sin((angleInDegrees - 90f) *
                Math.PI / 180F)) + origin.Y;
            return new PointF(x, y);
        }
    }
}
```

Custom Controls

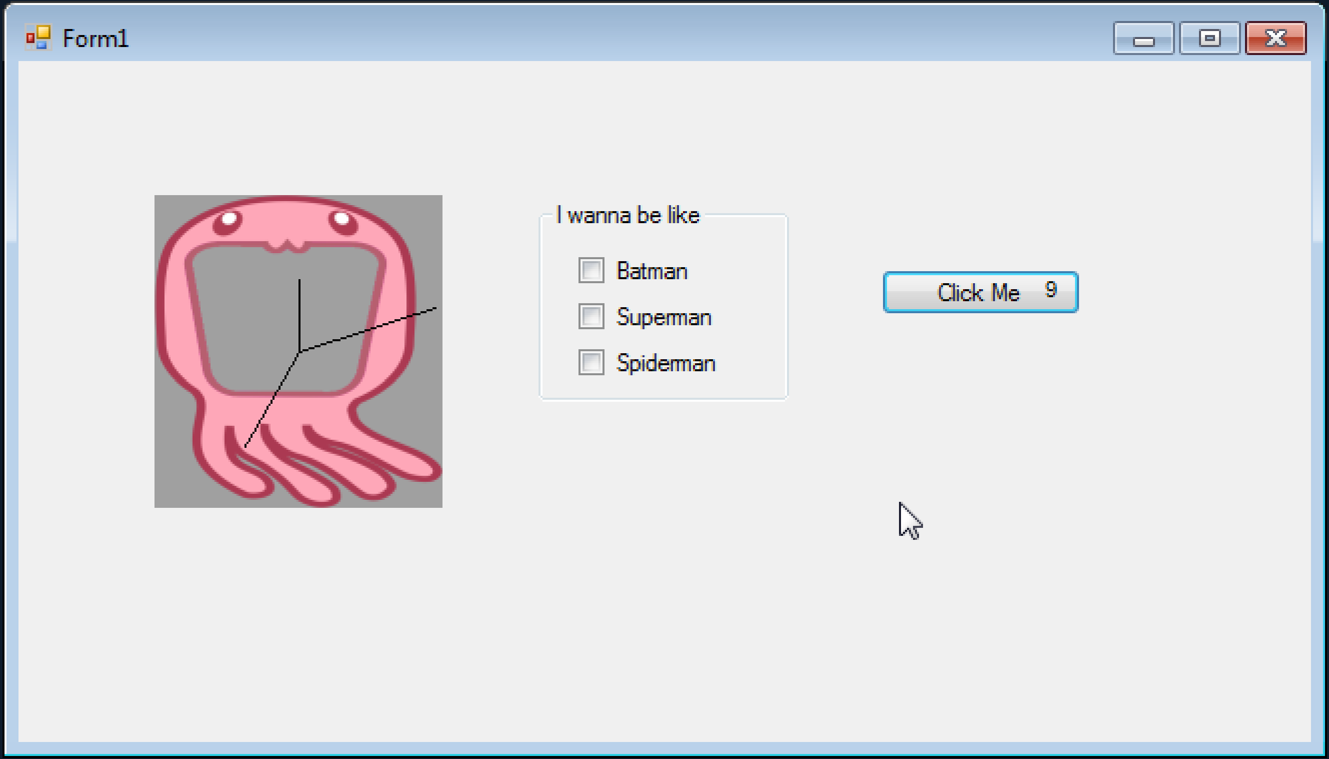
- Build the project
- Add a reference in your project containing the form to the library containing the new control
  - From the *Project* menu select *Add Reference*
- Add the new control to your form


Reference Manager - CustomControlForm

	Name	Path
✓	CustomControl	C:\AIE\2015\Subject_ADip_IntroductionToCS

Name: CustomControl

Custom Controls



Summary

- Composite Controls group standard controls together in one container
  - Easiest to make
- Extended Controls inherit from an existing control and extend functionality
- Custom Controls provide greatest flexibility, but most implementation done by us
  - Must override OnPaint method

References

- Microsoft Developer Network. 2015. *Varieties of Custom Controls*. [ONLINE] Available at:https://msdn.microsoft.com/en-us/library/ms171725(v=vs.110).aspx. [Accessed 28 January 15].
- Microsoft Developer Network. 2015. *How to: Author Composite Controls*. [ONLINE] Available at:https://msdn.microsoft.com/en-us/library/3sf86w5h(v=vs.110). [Accessed 28 January 15].
- Microsoft Developer Network. 2015. *How to: Inherit from Existing Windows Forms Controls*. [ONLINE] Available at: https://msdn.microsoft.com/en-us/library/7h62478z(v=vs.110).aspx. [Accessed 28 January 15].
- Microsoft Developer Network. 2015. *How to: Develop a Simple Windows Forms Control*. [ONLINE] Available at: https://msdn.microsoft.com/en-us/library/649xahhe(v=vs.110).aspx. [Accessed 28 January 15].