GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

DESARROLLO DE INTERFACES
UD05. CONTROLES PERSONALIZADOS

Departamento de Informática
Francisco Lifante Rivera

1. Introducción

Hasta ahora, solo hemos utilizado los controles integrados que se encuentran en el framework de WPF. Éstos nos darán un completísimo abanico de posibilidades, porque son extremadamente flexibles y pueden ser diseñados y preparados para hacer casi cualquier cosa. Sin embargo, en algún momento, es probable que tengáis que crear vuestros propios controles.

En otros UI frameworks, esto puede ser bastante engorroso, pero WPF lo hace bastante fácil, ofreciéndo dos formas de llevar a cabo esta tarea: mediante: **UserControls** y **Custom controls**.

UserControl

Un UserControl de WPF hereda la clase UserControl y se parece mucho a una ventana de WPF: tiene un archivo XAML y un archivo de código subyacente. En el archivo XAML, puedes agregar controles WPF que existen para crear el aspecto que desees y luego combinarlo con el código en el archivo de código subyacente, para lograr la funcionalidad deseada. De esta manera, WPF te permitirá integrar esta colección de funciones en uno o varios lugares de tu aplicación, permitiéndote agrupar y reutilizar de forma fácil funcionalidad en tus aplicaciones.

CustomControl

Un control personalizado es de más bajo nivel que un control de usuario. Cuando creas un control personalizado, heredas de una clase existente, dependiendo de cuanto quieras profundizar.

En muchos casos, puedes heredar la clase Control, de la cual otros controles WPF heredan (por ejemplo, el TextBox), pero si necesitas ir aún más allá, puedes heredar de FrameworkElement o incluso de UIElement. Cuanto más hacia abajo vayas, más control tendrás y menos funcionalidad será heredada.

El aspecto del custom control es controlado generalmente a través de los estilos en un archivo donde establecemos temas personalizados, mientras que el aspecto del User control seguirá el aspecto del resto de la aplicación. Esto también destaca una de las mayores diferencias entre un UserControl y un Custom control: el custom control puede ser diseñado/basado en plantilla, mientras que un UserControl no. En general, podríamos hablar de estos tres tipos de controles como diferentes capas de abstracción:

1) **Control personalizado:** un control personalizado, que se hereda directamente de la clase base *Control*, es útil si deseas un control completo sobre el aspecto y el comportamiento del control. Si bien puedes controlar la apariencia del control en el código, es más común definir un ControlTemplate que controle el diseño del control.
2) **Control de contenido:** un control de contenido hereda de la clase base *ContentControl* (que se deriva de Control). La clase ContentControl expone las propiedades Content y ContentTemplate que permiten al usuario especificar el contenido que se mostrará y la plantilla que se utilizará para mostrarlo, todo ello sin alterar la funcionalidad del control. El botón es un buen ejemplo de ContentControl: puedes especificar XAML como contenido o puedes establecer el contenido en datos y luego proporcionar una plantilla de contenido para definir cómo se deben mostrar los datos.
3) **Control de usuario:** un control de usuario hereda de la clase base *UserControl* (que se deriva de Control). La clase UserControl expone solo una propiedad de contenido que es un UIElement. A diferencia de ContentControl, que se basa en ContentTemplate para definir cómo se debe mostrar el contenido (los cuales pueden ser sobrescritos por el usuario del control), la propiedad Content de UserControl

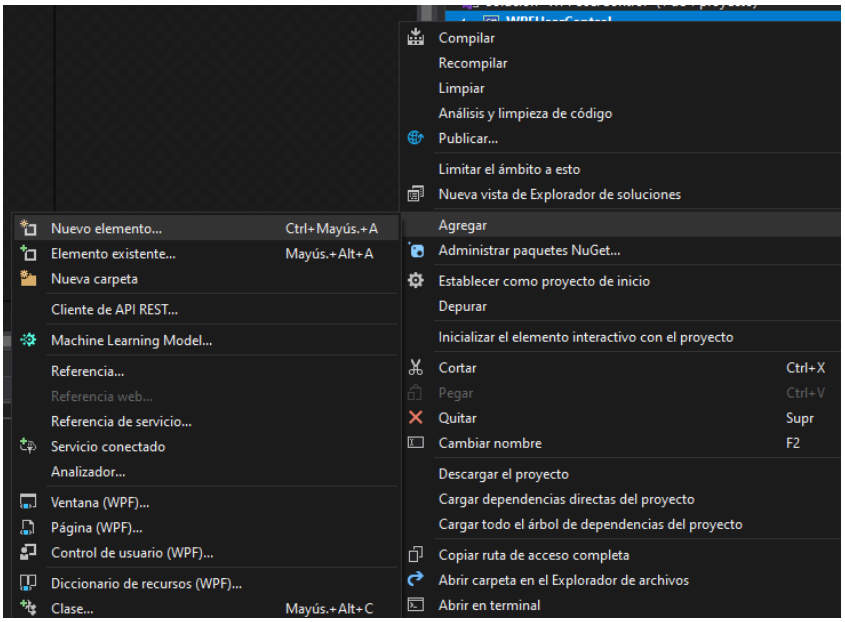
suele estar predefinida por el desarrollador del control. Esta es la forma más sencilla de crear un control y es útil para reutilizar fragmentos de XAML en una aplicación.

2. Control de usuario

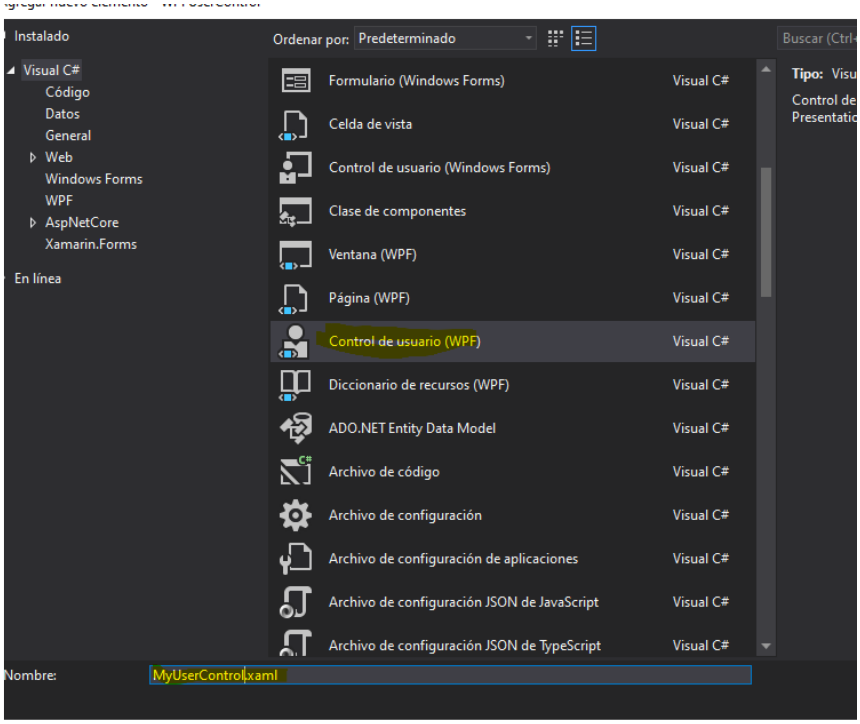
Una de las cosas para las que los controles de usuario son excelentes es para reutilizar fragmentos de XAML. Por ejemplo, pensemos que quiero proporcionar una interfaz donde el usuario pueda ingresar el nombre y apellido de alguien. Esta misma interfaz puede aparecer en varias páginas diferentes dentro de la aplicación, así que en lugar de simplemente copiar el XAML, podemos crear un control de usuario llamado Perfil (con la idea de que más adelante agregaremos otra información como la fecha de nacimiento y quizás alguna información de contacto).

Ejemplo control de usuario

Creamos un nuevo proyecto WPF y luego hacemos clic con el botón derecho en su solución y seleccionamos Agregar > Nuevo elemento…



Se abrirá la siguiente ventana. Ahora selecciona Control de usuario (WPF) y asígnale el nombre MyUserControl (por ejemplo...).



De esta forma se agregarán dos nuevos archivos a la solución. El .xaml y el .cs.

- El código XAML podría ser el siguiente:

```
<UserControl x:Class="WPFUserControl.MyUserControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:WPFUserControl"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <TextBox Height = "25"
         HorizontalAlignment = "Left"
         Margin = "80,49,0,0" Name = "txtBox"
         VerticalAlignment = "Top" Width = "200" />

        <Button Content = "Botón"
         Height = "25" HorizontalAlignment = "Left"
         Margin = "80,80,0,0" Name = "button"
         VerticalAlignment = "Top" Click = "button_Click" />
    </Grid>
</UserControl>
```

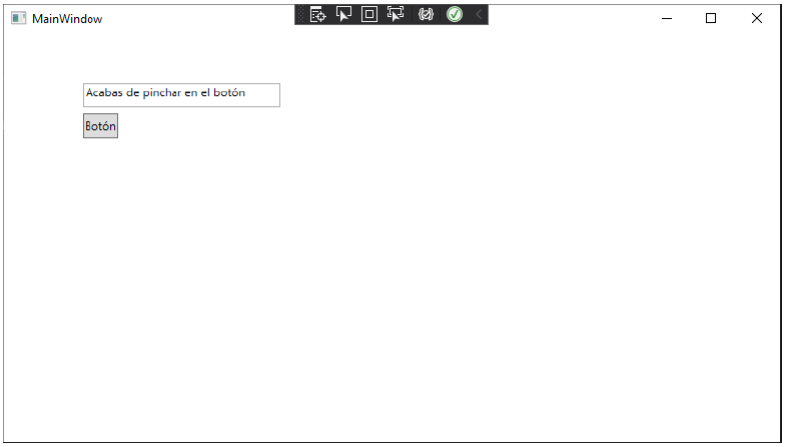
El código C# para el evento clic del botón que se encuentra en MyUserControl.cs que actualiza el cuadro de texto:

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace WPFUserControl
{
    /// <summary>
    /// Lógica de interacción para MyUserControl.xaml
    /// </summary>
    public partial class MyUserControl : UserControl
    {
        public MyUserControl()
        {
            InitializeComponent();
        }
        private void button_Click(object sender, RoutedEventArgs e)
        {
            txtBox.Text = "Acabas de pinchar en el botón";
        }

    }
}
```

La implementación en MainWindow.xaml para agregar el control de usuario:

```
<Window x:Class="WPFUserControl.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:control ="clr-namespace:WPFUserControl"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WPFUserControl"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <control:MyUserControl/>
    </Grid>
</Window>
```

La ventana resultante, en la cual al pinchar sobre el botón, se actualiza el texto dentro del cuadro de texto.



3. Controles personalizados

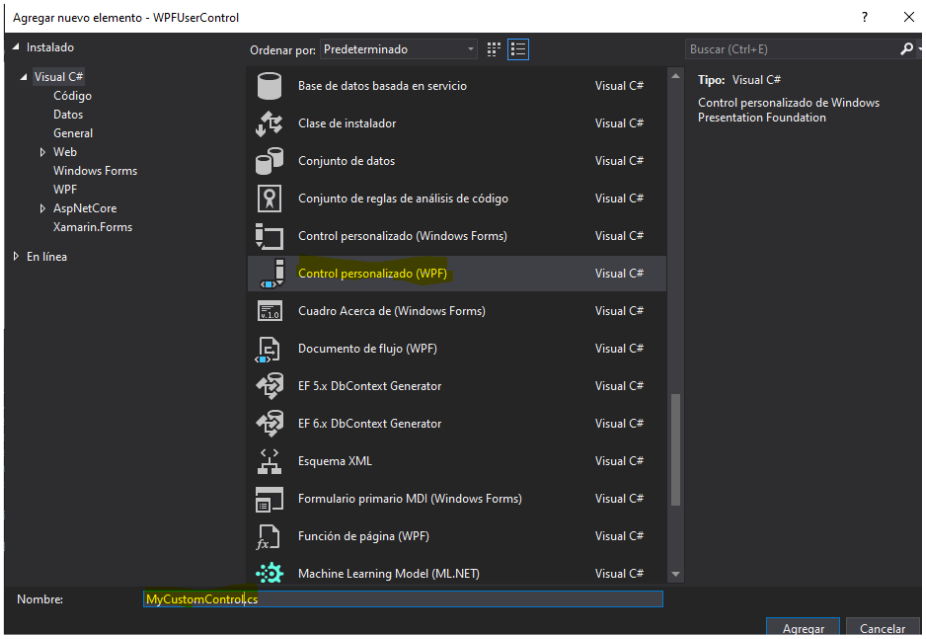
Un control personalizado es una clase que ofrece su propio estilo y plantilla que normalmente se definen en generic.xaml. Los controles personalizados se utilizan en los siguientes escenarios:

- ✓ Si el control no existe y hay que crearlo desde cero.
- ✓ Si deseas ampliar o agregar funcionalidad a un control preexistente agregando una propiedad o funcionalidad adicional para adaptarse a un escenario específico.
- ✓ Si tus controles necesitan admitir temas y estilos.
- ✓ Si deseas compartir el control entre aplicaciones.

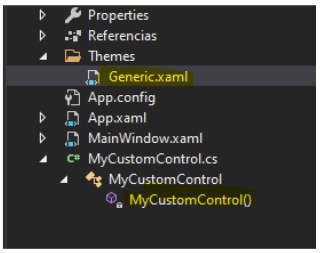
Ejemplo control personalizado

Al igual que con el control de usuario, pinchamos con el botón derecho en la solución y seleccionamos “agregar>nuevo elemento”.

Se abrirá la siguiente ventana en la que seleccionamos “control personalizado (WPF)” y le asignamos el nombre “MyCustomControl”



En esta ocasión se crean los siguientes 2 ficheros: Themes/Generic.xaml y MyCustomControl.cs.



Este es el código XAML en el que se establece el estilo para el control personalizado en el archivo Generic.xaml. (En este caso el color del fondo y la fuente del botón)

```
<ResourceDictionary
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local = "clr-namespace:WPFCustomControls">

    <Style TargetType = "{x:Type local:MyCustomControl}"
      BasedOn = "{StaticResource {x:Type Button}}">
        <Setter Property = "Background" Value = "#FF08D0F0" />
        <Setter Property = "Foreground" Value = "#FFF71C4E"/>
    </Style>

</ResourceDictionary>
```

Aquí está el código C# para la clase MyCustomControl que se hereda de la clase de botón y en el constructor anula los metadatos.

```
using System;
using System.Windows;
using System.Windows.Controls;


namespace WPFCustomControls
{

    public class MyCustomControl : Button
    {

        static MyCustomControl()
        {
            DefaultStyleKeyProperty.OverrideMetadata(typeof(MyCustomControl), new
               FrameworkPropertyMetadata(typeof(MyCustomControl)));
        }

    }
}
```

La implementación del evento de clic de control personalizado en C# que actualiza el texto del bloque de texto.

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace WPFCustomControls
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>

    public partial class MainWindow : Window
    {

        public MainWindow()
        {
            InitializeComponent();
        }

        private void customControl_Click(object sender, RoutedEventArgs e)
        {
            txtBlock.Text = "Has pinchado sobre un control personalizado";
        }
    }
}
```

Y la implementación en MainWindow.xaml para agregar el control personalizado y un TextBlock.

```
<Window x:Class = "WPFCustomControls.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:control = "clr-namespace:WPFCustomControls"
   Title = "MainWindow" Height = "350" Width = "604">

    <StackPanel>
        <control:MyCustomControl x:Name = "customControl"
         Content = "Click Me" Width = "70"
         Margin = "10" Click = "customControl_Click"/>

        <TextBlock Name = "txtBlock"
         Width = "250" Height = "30"/>
    </StackPanel>

</Window>
```

Al compilar y ejecutar el código anterior, se genera la siguiente ventana con un control personalizado que es un botón personalizado, en el cual, al pinchar, se actualiza el bloque de texto.