

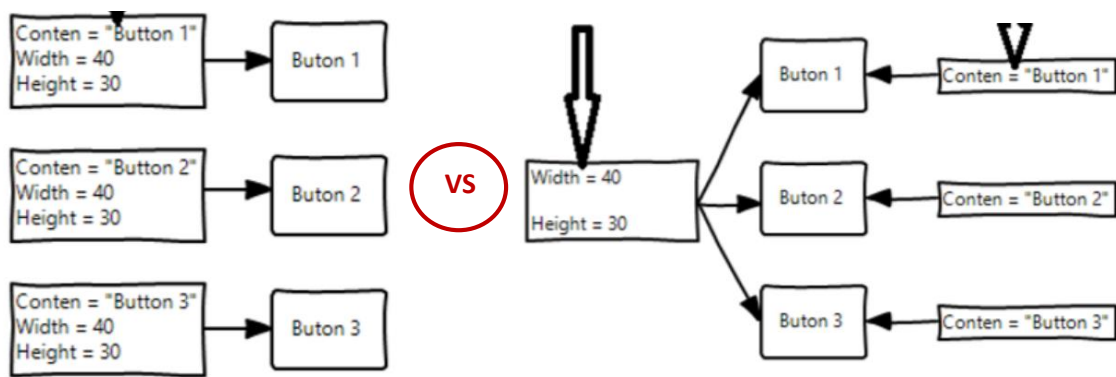
ESTILOS EN WPF

En las aplicaciones WPF, la estructura de los elementos de la interfaz de usuario de XAML se representa mediante un **árbol lógico**, es decir, sigue una estructura **jerárquica**.

Formar una estructura jerárquica implica que los estilos pueden ser definidos para diferentes ámbitos o niveles, dependiendo de dónde y cómo usarlos.

1. ESTILOS

Definir estilos permite obtener una apariencia consistente. Se flexibiliza el establecimiento de las propiedades de los objetos y permite reutilizar código, como por ejemplo, dar un aspecto uniforme a un conjunto de botones.



Los estilos son elementos del tipo **<Style>** que suelen declararse como **<Resources>**. Además, suelen definirse dentro de un **diccionario de recursos**.

Las propiedades se establecen mediante **<Setter>** junto con el nombre de la propiedad (**Property**) y su valor (**Value**).

Cada estilo tiene un **identificador de clave** único (x:Key) y un **tipo de destino** (TargetType).

Los estilos se pueden definir en los siguientes ámbitos o niveles:

- **Control**: solo aplica a un control en particular.
- **Diseño** (Layout): será accesible solo para un diseño y sus elementos secundarios.
- **Ventana**: todos los elementos de esa ventana puedan acceder al estilo.
- **Aplicación**: el estilo será accesible en toda la aplicación, en este caso, los estilos se suelen incluir en el archivo app.xaml.

Para aplicar un estilo que esté dentro de un diccionario de recursos, será necesario referirlo mediante **Style="{StaticResource NombreEstilo}"**. En este caso, la propiedad **Style** está implementada en la clase base **FrameworkElement**, por lo que todo elemento que herede de forma directa o indirecta de esta clase base adoptará la apariencia del estilo.

El estilo puede ejecutarse de dos maneras [StaticResource] o [DynamicResource]. La diferencia es que [DynamicResource] permite cambiar el estilo en tiempo de ejecución (code behind) frente a [StaticResource] que presentará el aspecto original.

A continuación, se propone un ejemplo de este último concepto donde se define una apariencia para los botones de una ventana y se establece el identificador clave "AparienciaBotones"

```
<Window.Resources>
  <Style x:Key="AparienciaBotones" TargetType="Button">
    <Setter Property="Background" Value="DarkRed" />
    <Setter Property="Foreground" Value="White" />
    <Setter Property="FontFamily" Value="Comic Sans MS" />
    <Setter Property="Height" Value="80"></Setter>
    <Setter Property="Width" Value="150"></Setter>
    <Setter Property="Margin" Value="20"></Setter>
    <Setter Property="HorizontalAlignment" Value="Left"></Setter>
    <Setter Property="VerticalAlignment" Value="Center"></Setter>
  </Style>
</Window.Resources>
```

Para utilizarlo en los botones que se añadan a la ventana:

```
<Button Style="{StaticResource AparienciaBotones}"
  x:Name="btnEjemplo1" Content="Ejemplo botones iguales" />
<Button Style="{StaticResource AparienciaBotones}"
  x:Name="btnEjemplo2" Content="Ejemplo diferente"
  Background="DarkOliveGreen"
  HorizontalAlignment="Right"
  Margin="50, 50, 0, 0" />
```

El segundo botón incorpora estilos propios.

2. ÁMBITOS O NIVELES

Los ejemplos están disponibles en el archivo EjemploEstilos.zip.



Estilos de control


A nivel de elemento:

```
<!--Estilo individual para un botón-->
<Button x:Name="btnEstiloIndividual" Content = "Estilo individual" Margin="10">
  <Button.Resources>
    <Style TargetType="{x:Type Button}">
      <Setter Property = "Width" Value = "100" />
      <Setter Property = "Height" Value = "40" />
      <Setter Property = "Margin" Value = "50" />
    </Style>
  </Button.Resources>
</Button>
<!--Alternativa de estilo individual para un botón-->
<Button x:Name="btnAlternativaIndividual" Content = "Alternativa estilo individual" Margin="10">
  <Button.Style>
    <Style>
      <Setter Property = "FrameworkElement.Width" Value = "200" />
      <Setter Property = "FrameworkElement.Height" Value = "40" />
      <Setter Property = "FrameworkElement.Margin" Value = "50" />
    </Style>
  </Button.Style>
</Button>
```

Estilos de diseño

Para agrupar los estilos de los elementos contenidos dentro de <Grid>, <StackPanel>, etc.

```
<!--Estilo a nivel de diseño-->
<StackPanel Margin = "10">
    <StackPanel.Resources>
        <Style TargetType = "{x:Type Button}">
            <Setter Property = "Foreground" Value =  "White" />
            <Setter Property = "Background" Value =  "Orange" />
            <Setter Property = "FontStyle" Value = "Italic" />
            <Setter Property = "Width" Value = "100" />
            <Setter Property = "Height" Value = "20" />
            <Setter Property = "Margin" Value = "10" />
        </Style>
    </StackPanel.Resources>

    <Button Content="Button 1"/>
    <Button Content="Button 2"/>
    <Button Foreground =  "Blue" Content="Button 3"/>
</StackPanel>
```

Estilos de ventana

En la cabecera, detrás del cierre de la etiqueta <Window>, se añade una etiqueta <Window.Resources>, puedes crear un identificador de clave o no, depende de tu diseño. En el ejemplo no se ha creado, solo se ha indicado el destino del estilo:

```
<!--Definición de estilos a nivel de ventana-->
<Window.Resources>
    <Style TargetType = "TextBlock">
        <Setter Property = "FontSize" Value = "24" />
        <Setter Property = "Margin" Value = "40,0,0,0" />
        <Setter Property = "FontWeight" Value = "Bold" />
    </Style>

    <Style TargetType = "TextBox">
        <Setter Property = "HorizontalAlignment" Value = "Left" />
        <Setter Property = "FontSize" Value = "24" />
        <Setter Property = "Margin" Value = "5" />
        <Setter Property = "Width" Value = "200" />
        <Setter Property = "Height" Value = "40" />
    </Style>
</Window.Resources>
```

Para comprobarlo, añadimos un <Grid> con elementos <TextBox> y <TextBlock>

```

<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height = "Auto" />
    <RowDefinition Height = "Auto" />
    <RowDefinition Height = "*" />
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
    <ColumnDefinition Width = "*" />
    <ColumnDefinition Width = "2*" />
  </Grid.ColumnDefinitions>

  <TextBlock Text = "Nombre: " />
  <TextBox x:Name = "txtNombre" Grid.Column = "1" />
  <TextBlock Text = "Apellidos: " Grid.Row = "1" />
  <TextBox x:Name = "txtApellidos" Grid.Column = "1" Grid.Row = "1" />
</Grid>

```

Estilos de aplicación

En el App.xml, dentro de la etiqueta <Application.Resources>:

```

<Application.Resources>
  <!--Estilos de aplicación-->
  <Style TargetType = "TextBlock">
    <Setter Property = "FontSize" Value = "24" />
    <Setter Property = "Margin" Value = "40,0,0,0" />
    <Setter Property = "FontWeight" Value = "Bold" />
  </Style>

  <Style TargetType = "TextBox">
    <Setter Property = "HorizontalAlignment" Value = "Left" />
    <Setter Property = "FontSize" Value = "24" />
    <Setter Property = "Margin" Value = "5" />
    <Setter Property = "Width" Value = "200" />
    <Setter Property = "Height" Value="40" />
  </Style>
</Application.Resources>

```

Para probar la configuración, agregamos un nuevo elemento de tipo ventana WPF.

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height = "Auto" />
    <RowDefinition Height = "Auto" />
    <RowDefinition Height = "*" />
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
    <ColumnDefinition Width = "*" />
    <ColumnDefinition Width = "2*" />
  </Grid.ColumnDefinitions>

  <TextBlock Text = "Nombre: " />
  <TextBox x:Name = "txtNombre" Grid.Column = "1" />
  <TextBlock Text = "Apellidos: " Grid.Row = "1" />
  <TextBox x:Name = "txtApellidos" Grid.Column = "1" Grid.Row = "1" />
</Grid>
```