Estilos en WPF

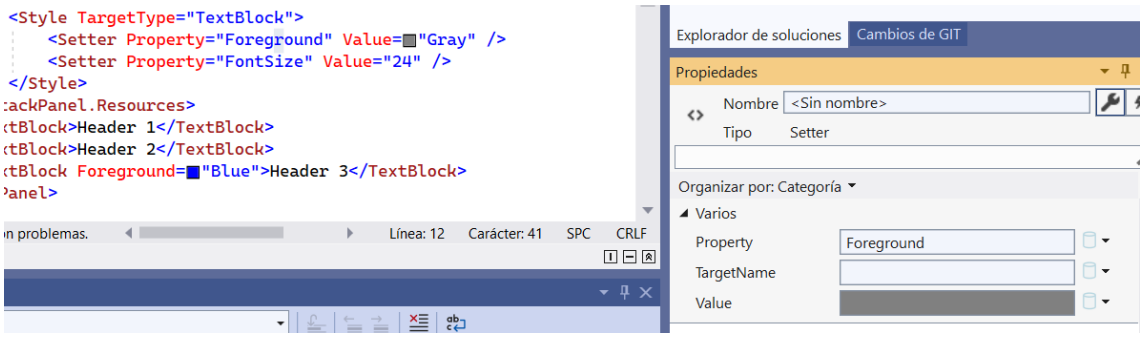
WPF introduce estilo, que es para XAML lo que CSS es para HTML. Usando estilos, se pueden agrupar un conjunto de propiedades y asignarlas a controles específicos o a todos controles de un tipo específico, y al igual que en CSS, un estilo puede heredar de otro estilo.

Es posible crear los estilos como recursos, ya sea a nivel de elemento, ventana o aplicación.

```
<StackPanel Margin="10">
  <StackPanel.Resources>
    <Style TargetType="TextBlock">
      <Setter Property="Foreground" Value="Gray" />
      <Setter Property="FontSize" Value="24" />
    </Style>
  </StackPanel.Resources>
  <TextBlock>Header 1</TextBlock>
  <TextBlock>Header 2</TextBlock>
  <TextBlock Foreground="Blue">Header 3</TextBlock>
</StackPanel>
```

Cada propiedad susceptible de tener un estilo en un control se puede crear en la etiqueta **Style.** El atributo **TextBlock** indica para que tipo de control se aplicará. Luego cada Setter se corresponde con una propiedad válida en este caso de un TextBlock.

Además, el editor gráfico nos permite seleccionar entre los valores posibles de cada propiedad.



La visualización final será la siguiente. El último texto tiene color azul porque se asigna explícitamente y tiene prioridad respecto a aquellos estilos de la etiqueta **Style**. Básicamente como ocurre con HTML y CSS.

Header 1
Header 2
Header 3

Es posible combinar varios estilos. Por ejemplo, podemos definir algunos a nivel de ventana para el mismo recurso.

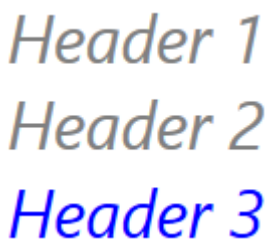
```
<Window.Resources>
  <Style x:Key="estiloText" TargetType="TextBlock">
    <Setter Property="FontStyle" Value="Italic" />
  </Style>
</Window.Resources>
```

Pero por defecto no funciona. La forma de garantizar que se apliquen ambos estilos es, por un lado, añadiendo el atributo **x:Key** como en un recurso.

Luego el estilo anterior podemos hacer que se base en el anterior empleando el **x:Key** anterior y así sucesivamente encadenando tantos estilos como queramos.

```
<StackPanel.Resources>
  <Style TargetType="TextBlock" BasedOn="{StaticResource estiloText}">
    <Setter Property="Foreground" Value="Gray" />
    <Setter Property="FontSize" Value="24" />
  </Style>
</StackPanel.Resources>
```

El resultado que combinaría todos los estilos sería el siguiente:



En el caso anterior hemos utilizado el estilo creado en Window a partir del **BasedOn**. Es posible definir estilos con un **x:Key** y definirlos explícitamente.

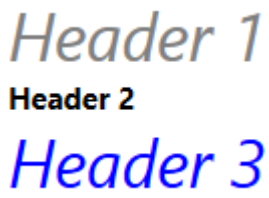
Ahora vamos a probar en el **App.xaml** y que de paso puedas observar que podemos definir estilos para varias ventanas.

```
<Application.Resources>
  <Style x:Key="boldStyle" TargetType="TextBlock">
    <Setter Property="FontWeight" Value="Bold" />
  </Style>
</Application.Resources>
```

Este estilo solo se podría aplicar explícitamente indicando el **x:Key** en el atributo **Style** del ítem que corresponda.

```
<TextBlock Style="{StaticResource boldStyle}">Header 2</TextBlock>
```

La visualización quedaría como se indica debajo.



Aunque como puedes observar han desaparecido los estilos anteriores. Para combinarlos tendríamos que encadenar las etiquetas con **BasedOn**. Si se emplea un estilo aparte con un **x:Key** como hemos hecho, no se combina y sobrescribe aquellos con menor prioridad.