GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

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

DESARROLLO DE INTERFACES
UD05. CONTROL TEMPLATE Y TRIGGERS

Departamento de Informática
Luis José Fortich Giner

ControlTemplate

Es una parte esencial de la arquitectura de diseño que permite definir la apariencia visual y la estructura de presentación de un control. En términos sencillos, un ControlTemplate describe cómo se renderiza un control visualmente

Propósito del ControlTemplate:

- **Separación de la Lógica y Presentación**
  Facilita la separación entre la lógica de un control y su presentación visual. Esto sigue el principio de separación de preocupaciones (Separation of Concerns) en el desarrollo de software, donde la lógica y la presentación están claramente definidas y separadas.
- **Personalización y Estilo**
  Permite personalizar la apariencia de los controles sin tener que cambiar la lógica subyacente. Puedes redefinir cómo se ve un control sin afectar su comportamiento funcional.
- **Reutilización**
  Facilita la reutilización del código y la apariencia. Puedes definir un ControlTemplate para un tipo de control y reutilizarlo en múltiples lugares de tu aplicación.

Estructura de un ControlTemplate:

Un ControlTemplate está definido en XAML y puede contener una estructura compleja que describe cómo se deben visualizar los elementos dentro del control.

Ejemplo simple de un ControlTemplate para un botón:

```
<ControlTemplate TargetType="Button">
  <Border Background="{TemplateBinding Background}"
      BorderBrush="{TemplateBinding BorderBrush}"
      BorderThickness="1">
    <ContentPresenter HorizontalAlignment="Center"
          VerticalAlignment="Center"/>
  </Border>
</ControlTemplate>
```

Uso en XAML

Cuando defines un ControlTemplate, generalmente lo asignas a un control específico utilizando la propiedad Template del control. Por ejemplo, para aplicar el ControlTemplate de botón anterior:

```
<Button Content="Haz clic" Template="{StaticResource TuControlTemplate}" />
```

Diferencias entre ControlTemplate y Style

Ambos son herramientas que nos permiten personalizar la apariencia de los controles pero tienen propósitos ligeramente diferentes y se utilizan en contextos diferentes.

Mientras que el ***ControlTemplate*** define la estructura visual completa de un control, incluidos sus elementos internos y su disposición, los **estilos** se utilizan para establecer propiedades de presentación tales como colores, fuentes, márgenes, etc. Realiza ajustes visuales sin realizar cambios en la estructura interna del control.

El ***ControlTemplate*** se utiliza cuando necesitas cambiar completamente la apariencia y estructura de un control. Por ejemplo, para crear un botón completamente personalizado con una estructura diferente, añadir animaciones, etc. Por su parte, los **estilos** se utilizan para aplicar ajustes de presentación como colores, fuentes, márgenes, etc.

Resumiendo:

Estructura vs. Presentación:

ControlTemplate se enfoca en definir la estructura visual del control, mientras que el ***Style*** se enfoca en definir las propiedades de presentación.

Cambios en la Estructura:

- ***ControlTemplate*** te permite realizar cambios significativos en la estructura interna del control. Puedes redefinir completamente cómo se ve y se compone el control.
- ***Style*** generalmente se utiliza para cambios en la presentación sin cambiar la estructura interna del control.

Flexibilidad:

- ***ControlTemplate*** es más flexible y puede realizar cambios radicales en la apariencia y el diseño del control.
- ***Style*** es más adecuado para cambios más simples y ajustes visuales.

En muchos casos, se pueden combinar ambos enfoques. Puedes aplicar un ***ControlTemplate*** personalizado para cambiar la estructura principal y luego aplicar un ***Style*** para ajustar propiedades específicas de presentación. La elección entre uno u otro depende de la complejidad de los cambios que necesitas realizar en la apariencia del control.

DataTrigger

Un DataTrigger es un tipo de disparador en XAML que se utiliza en tecnologías como WPF y Xamarin.Forms. Los DataTriggers permiten cambiar el aspecto visual de un elemento basándose en el valor de una propiedad de datos. Se activan cuando la propiedad de datos alcanza un valor específico.

Si miráis el código del ejemplo DataTrigger en la solución “PersonalizacionControles” subido a Aules, podemos ver lo siguiente:

Los DataTriggers se aplican al estilo de un elemento Label. Cada DataTrigger está asociado a un RadioButton específico (rdoRed, rdoGreen, rdoBlue). Cuando uno de estos botones de radio está seleccionado (IsChecked es True), se activa el correspondiente DataTrigger, lo que lleva a la aplicación de los cambios visuales definidos en el DataTrigger.

En resumen, los DataTriggers son útiles para crear interfaces de usuario dinámicas que responden a cambios en los datos. Pueden ser utilizados para cambiar propiedades visuales, animaciones u otros aspectos de la interfaz de usuario en función de las condiciones de los datos

en tiempo de ejecución. Esto es especialmente útil en escenarios donde deseas que la interfaz de usuario se adapte y cambie según el estado o las propiedades de los datos en tu aplicación.