Listas de datos

UT5. GENERACIÓN DE INTERFACES A TRAVES DE XML

Listas de datos

Existen múltiples controles para mostrar colecciones de datos.

- ItemsControl -> muy sencillo, no permite interactuar con la información.
- ListBox -> muy sencillo, formato mínimo, interacción mínima.
- ComboBox -> aunque se haya incluido en controles comunes es una lista.
- ListView -> altas capacidades de formato e interacción.
- DataGrid -> muy sencillo de usar, permite personalizar, los datos se editan sobre el propio control por lo que su uso no es aconsejable en la mayoría de los casos.

Vamos a centrarnos en el ListView por ser el control más habitual a la hora de presentar datos.

ListView

Permite mostrar la información con distintos tipos de formato.

- En WinForms el formato lo marca un enumerado, en WPF deberemos codificar de una u otra forma según el formato que deseemos.
- 3 aproximaciones iniciales:
  - Lista sin ningún formato.
  - Definición de GridView con columnas para el formato tabla.
  - Definición de ItemTemplate para un formato completamente personalizado.

Etiquetas principales.

- **ListView**, contenedor principal.
- **ListViewItem**, fila en el contenedor, envuelve un dato.
  - IsSelected, indica si la fila esta seleccionada.

ListView - II

Etiquetas formato tabla.

- ListView.View, únicamente hay un tipo hijo predefinido, el GridView.
  - **GridView**.
    - **GridViewColumn**, permite definir una columna en la vista. Si se desea se le puede definir una plantilla para la celda.
      - Header, texto de la columna de cabecera.
      - DisplayMemberBinding, nombre de la propiedad a la que enlazar.

Etiquetas formato personalizado.

- ListView.ItemTemplate, permite definir una plantilla para los ListViewItem
  - DataTemplate, contenedor de la plantilla.

ListView - III

Fuentes de datos

- **ItemsSource**, datos en el control, para cuando los gestionamos de manera automática.
- **Items**, datos en el control, para cuando los gestionamos de manera manual.

NOTA: no podemos combinar “ItemsSource” e “Items” a la vez.

```
List<Usuario> items = new List<Usuario>();

items.Add(new Usuario() { Nombre = "Pepe" });

lvUsuarios.ItemsSource = items;
```

O

```
lvUsuarios.Items.Add(new Usuario() { Nombre = "Pepe" });
```

ListView sin formato

```
<ListView Margin="10">
      <ListViewItem>Elemento #1</ListViewItem>
      <ListViewItem IsSelected="True">Elemento #2</ListViewItem>
      <ListViewItem>Elemento #3</ListViewItem>
</ListView>
```

ListView con formato personalizado

```
<ListView Margin="10" Name="lvUsuarios">
   <ListView.ItemTemplate>
      <DataTemplate>
         <WrapPanel>
            <TextBlock Text="Name: " />
            <TextBlock Text="{Binding Nombre}" FontWeight="Bold" />
            <TextBlock Text=" - " />
            <TextBlock Text="{Binding Correo}" TextDecorations="Underline"
Foreground="Blue"/>
         </WrapPanel>
      </DataTemplate>
   </ListView.ItemTemplate>
</ListView>
```

ListView con formato de tabla

```
<ListView Margin="10" Name="lvUsuarios">
    <ListView.View>
        <GridView>
            <GridViewColumn Header="Nombre" Width="120" DisplayMemberBinding="{Binding Nombre}"
/>
            <GridViewColumn Header="Correo" Width="150">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBlock Text="{Binding Correo}" TextDecorations="Underline"/>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
        </GridView>
    </ListView.View>
</ListView>
```

ListView - Ordenar

La colección .Items.SortDescriptions contiene una colección de criterios de ordenación.

Los criterios de ordenación son del tipo “SortDescription” que toma como parámetros el GridViewColumnHeader a ordenar y el ListSortDirection con el sentido.

- Ordenamos al hacer click sobre la cabecera de la columna

Recuerda limpiar los criterios de ordenación existente antes de establecer uno nuevo .Items.SortDescriptions.Clear()

ListView - Filtrar

Crear filtros es muy fácil, únicamente tenemos que localizar el objeto de la vista y asignarle una función con las condiciones de filtrado.

```
CollectionView vista = (CollectionView)CollectionViewSource.GetDefaultView(lvUsuarios.Items);

vista.Filter = FiltroVista;
```

La función de filtrado recibe un objeto y si devuelve cierto se muestra, fíjate que podemos establecer un filtro con múltiples propiedades.

```
private bool FiltroVista(object item)
```

NOTA: si el filtro no se actualiza reasigna la función de filtrado para forzar el refresco.