



Recursos y Estilos

Ricardo Vasquez Sierra

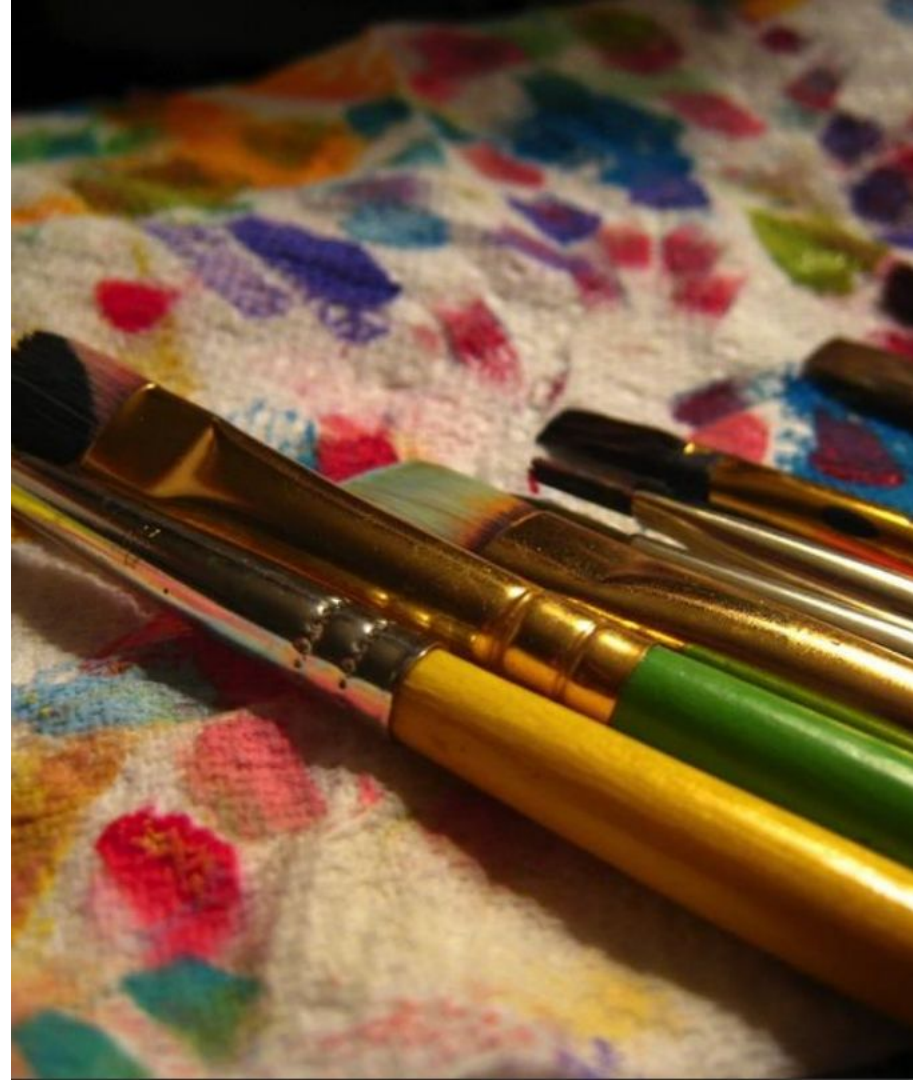
www.ricardovasquez.postach.io



Microsoft
CERTIFIED
Professional

Agenda

- ❖ Utilizar recursos a nivel de página
- ❖ Utilizar recursos globales
- ❖ Actualizar recursos dinámicamente



Motivación

- ❖ Eliminar valores duplicados de archivos XAML ya que son difíciles de mantener.
- ❖ Utilizar distintos temas dentro de cualquier aplicación.

```
<StackLayout BackgroundColor="#FFFFFF">  
  <Label      TextColor="#00FF00" FontSize="16.5" ... />  
  <Entry      BackgroundColor="#FFFFFF" ... />  
  <BoxView    BackgroundColor="#00FF00" ... />  
  <Button     BackgroundColor="#00FF00" FontSize="16.5" ... />  
</StackLayout>
```

Common to use the same colors and sizes across the UI

Diccionario de Recursos

- ❖ Es un recurso llave/valor personalizado para utilizarse con recursos de UI.

Mostly has
standard
dictionary
operations



```
public sealed class ResourceDictionary : ...  
{ ...  
    public object this[string index] { get; set; }  
  
    public void Add(string key, object value);  
    public void Add(Style implicitStyle);  
}
```



Some added UI-specific functionality

Recursos a Nivel de Página

- ❖ Todas las páginas pueden tener un diccionario de recursos y puede crearse desde XAML/C#.

You must create
the dictionary

```
<ContentPage ...>  
    <ContentPage.Resources>  
        <ResourceDictionary>  
            ...  
        </ResourceDictionary>  
    </ContentPage.Resources>  
</ContentPage>
```

Assign the dictionary
you create to the page's
Resources property

Creando Recursos Estáticos

- ❖ Los recursos creados desde XAML deben de utilizar la propiedad XAML **x:Key** para darle visibilidad dentro de la página.

Create inside
the page's
dictionary

```
<ContentPage ...>  
    <ContentPage.Resources>  
        <ResourceDictionary>  
            → <Thickness x:Key="myKey">10,20,40,80</Thickness>  
        </ResourceDictionary>  
    </ContentPage.Resources>  
</ContentPage>
```

Value

Key

Utilizando Recursos Estáticos

- ❖ Utilizando la extensión de marcado **{StaticResource}** se obtiene el recurso y es aplicado a un elemento determinado cuando este es creado.

```
<ContentPage ...>

    <ContentPage.Resources>
        <ResourceDictionary>
            Define → <Thickness x:Key="myKey">10,20,40,80</Thickness>
        </ResourceDictionary>
    </ContentPage.Resources>

    Use → <StackLayout Padding="{StaticResource myKey}">
        ...
    </StackLayout>

</ContentPage>
```

Dependencias de Plataforma

- ❖ Es posible usar **Device.OnPlatform** dentro de los diccionarios de recursos para manejar valores específicos de cada plataforma.

```
<ResourceDictionary>
  <OnPlatform x:Key="textColor"
    x:TypeArguments="Color"
    iOS="Silver"
    Android="Green"
    WinPhone="Blue" />
</ResourceDictionary>
```

```
<Label TextColor="{StaticResource textColor}" ... />
```


Demostración

Utilizando recursos a nivel de página



Recursos Dinámicos

- ❖ Recursos que pueden ser actualizados en tiempo de ejecución. Útiles en casos donde el usuario puede seleccionar diferentes valores.

Define a
default
in XAML

```
<ResourceDictionary>  
→ <Color x:Key="bg">Blue</Color>  
</ResourceDictionary>
```

Update
to new
value

```
void OnChangeColor()  
{  
→ this.Resources["bg"] = Color.Green;  
}
```

Utilizando Recursos Dinámicos

- ❖ La extensión de marcado **{DynamicResource}** obtiene un recurso cuando un elemento es creado y lo actualiza a medida que valor del mismo cambia.

BackgroundColor
set to **Blue** initially



```
<ResourceDictionary>
  <Color x:Key="bg">Blue</Color>
</ResourceDictionary>

<StackLayout BackgroundColor="{DynamicResource bg}">
  ...
</StackLayout>
```

BackgroundColor
changes to **Green**



```
void OnChangeColor()
{
  this.Resources["bg"] = Color.Green;
}
```

Aplicando Recursos Dinámicamente

- ❖ Los recursos pueden ser aplicados desde código utilizando el método **SetDynamicResource**.

```
var name = new Label { Text = "Name" };  
  
if (Device.OS == TargetPlatform.iOS)  
{  
    name.SetDynamicResource(Label.TextColorProperty, "hlColor");  
}
```

The **BindableProperty** to assign

The Resource key to apply

Demostración

Utilizando recursos dinámicos



Estilos

- ❖ Mejoran la reutilización de código al retirar la necesidad de establecer cada propiedad individualmente en cada instancia que es creada.

The property settings must be repeated on each view

```
<Button
  BackgroundColor="{StaticResource highlightColor}"
  BorderColor     ="{StaticResource edgeColor}"
  BorderRadius    ="{StaticResource edgeRadius}"
  BorderWidth     ="{StaticResource edgeSize}"
  TextColor       ="{StaticResource textColor}"
  Text             ="OK" />
```

OK

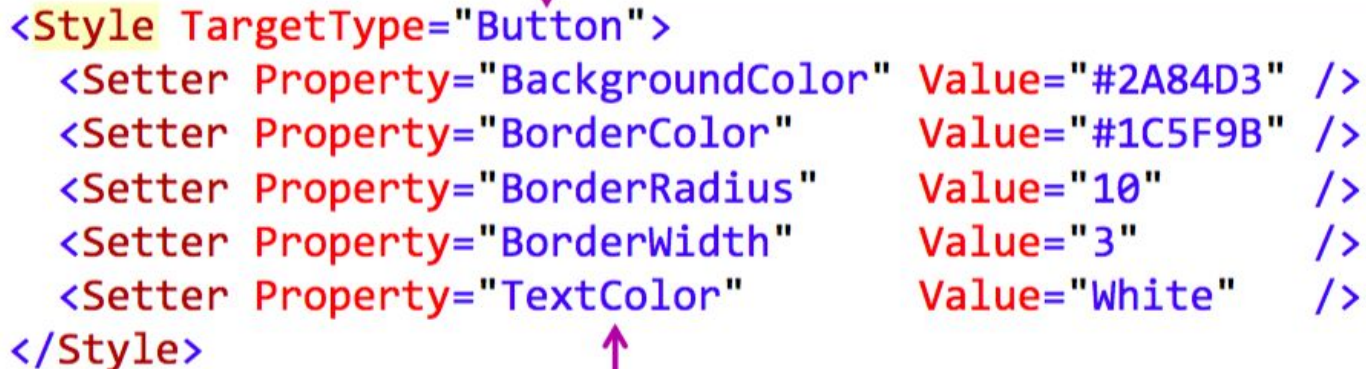
```
<Button
  BackgroundColor="{StaticResource highlightColor}"
  BorderColor     ="{StaticResource edgeColor}"
  BorderRadius    ="{StaticResource edgeRadius}"
  BorderWidth     ="{StaticResource edgeSize}"
  TextColor       ="{StaticResource textColor}"
  Text             ="Cancel" />
```

Cancel

Estructura de un Estilo

- ❖ Colección de **Setters** para un tipo de vista específica por lo general definidos dentro de un diccionario de recursos.

TargetType must be set (or runtime exception)



```
<Style TargetType="Button">
  <Setter Property="BackgroundColor" Value="#2A84D3" />
  <Setter Property="BorderColor" Value="#1C5F9B" />
  <Setter Property="BorderRadius" Value="10" />
  <Setter Property="BorderWidth" Value="3" />
  <Setter Property="TextColor" Value="White" />
</Style>
```

The properties must be members of the
TargetType class (or runtime exception)

Utilizando un Estilo

- ❖ Los estilos pueden ser asignados a un control en específico a través de la propiedad **Style**, la cual aplica todos los **Setters** del estilo al mismo.

```
<Button Text="OK" Style="{StaticResource MyButtonStyle}" />  
<Button Text="Cancel" Style="{StaticResource MyButtonStyle}" />
```



The **Style** property is defined
in the **VisualElement** base
class so it is available in all views


Combinando Estilos y Recursos

- ❖ Un recurso puede ser utilizado como valor para un setter, lo que le permite compartir su valor con otros estilos.

```
<Color x:Key="bgColor">White</Color>
<Color x:Key="fgColor">Black</Color>

<Style TargetType="Button" x:Key="AllButtons">
  <Setter Property="BackgroundColor" Value="{StaticResource bgColor}" />
  <Setter Property="TextColor" Value="{DynamicResource fgColor}" />
  ...
</Style>
```

Can use either static or dynamic lookup



Recursos Implícitos

- ❖ Si a un estilo le retiramos la propiedad **Key**, dicho estilo será aplicado a todos los controles que pertenezcan al tipo especificado en **TargetType**.

```
<ContentPage.Resources>
  <ResourceDictionary>
    <Style TargetType="Button">
      <Setter Property="BackgroundColor" Value="Blue" />
      <Setter Property="BorderColor" Value="Navy" />
      ...
    </Style>
  </ResourceDictionary>
</ContentPage.Resources>
```

The target type is still specified and is matched exactly,
this style will be applied to all buttons in this page

Herencia de Estilos

- ❖ Un estilo puede heredar de otro, y sobrescribir propiedades del estilo base así como añadir nuevas. Para aplicar herencia de estilos se utiliza la propiedad **BasedOn** en el nuevo estilo.

Base's **TargetType** must be the same or a base class

```
<Style x:Key="MyButtonStyle" TargetType="Button">
  ...
</Style>

<Style x:Key="DiscoButtonStyle" TargetType="Button" BasedOn="{StaticResource MyButtonStyle}">
  ...
</Style>
```

Indicates which style
this will inherit from

Only **StaticResource** is
allowed to set the base style

Demostración

Utilizando estilos





Gracias! :)

Referencias:

Xamarin University



Microsoft
CERTIFIED
Professional