Renderizadores

Ricardo Vasquez Sierra

www.ricardovasquez.postach.io

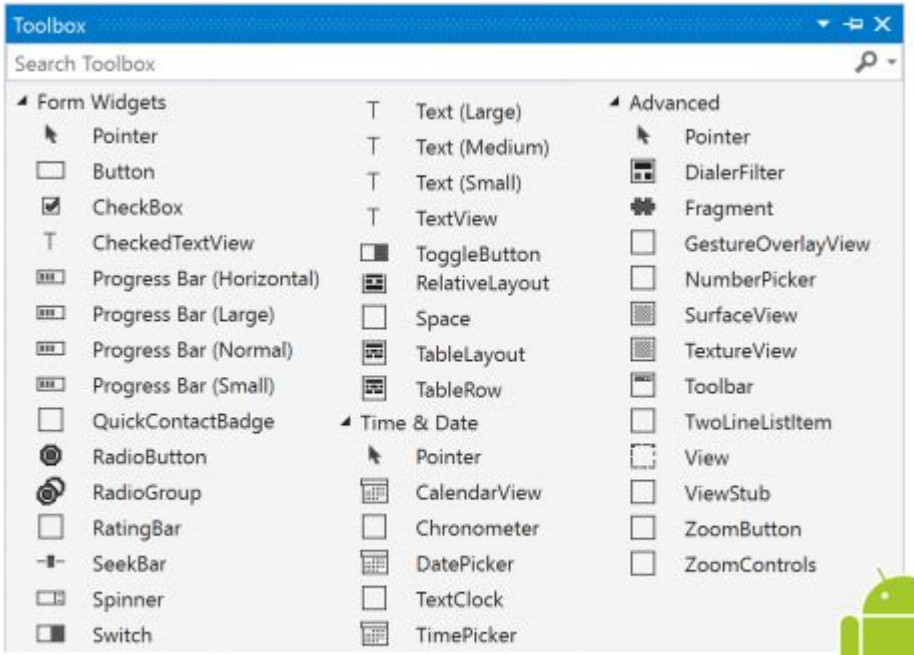
Microsoft CERTIFIED Professional

Agenda

- Añadir controles nativos a Xamarin.Forms
- Personalizar renderizador de un control existente
- Crear un renderizador para un control personalizado de Xamarin.Forms

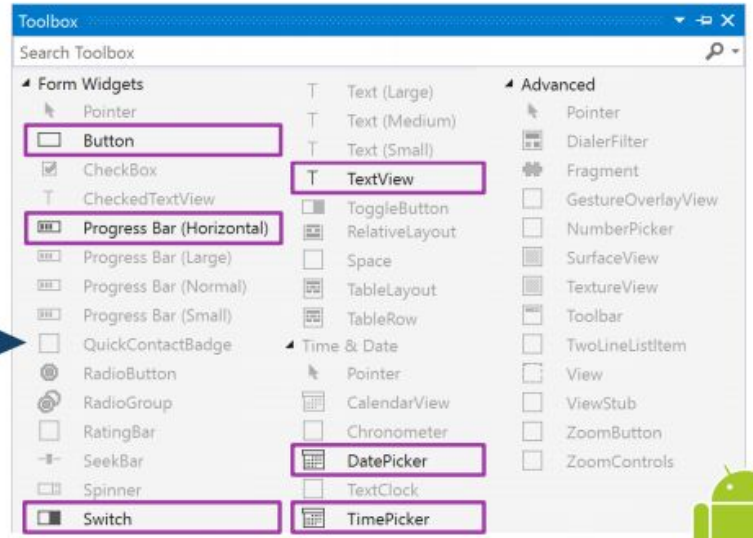
¿Que es un Control Nativo?

- Un control nativo es lo que presenta la UI en las aplicaciones.
- Cada plataforma tiene su propia selección de controles, de los cuales, muchos son únicos en cada una.



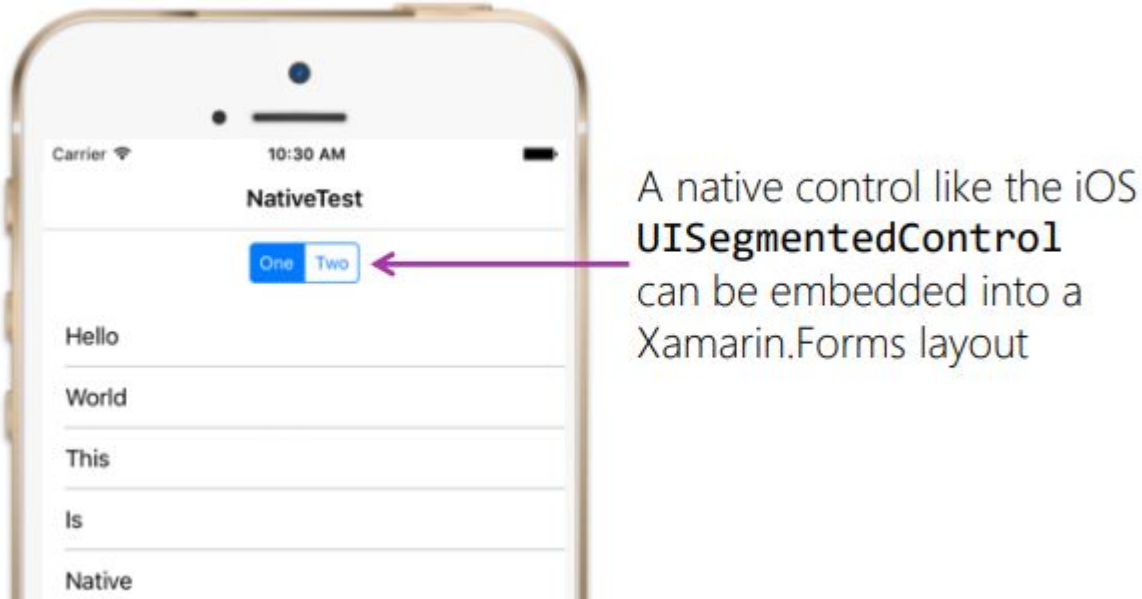
Set de Controles en Xamarin.Forms

- Xamarin.Forms expone un conjunto común de controles en todas las plataformas soportadas.



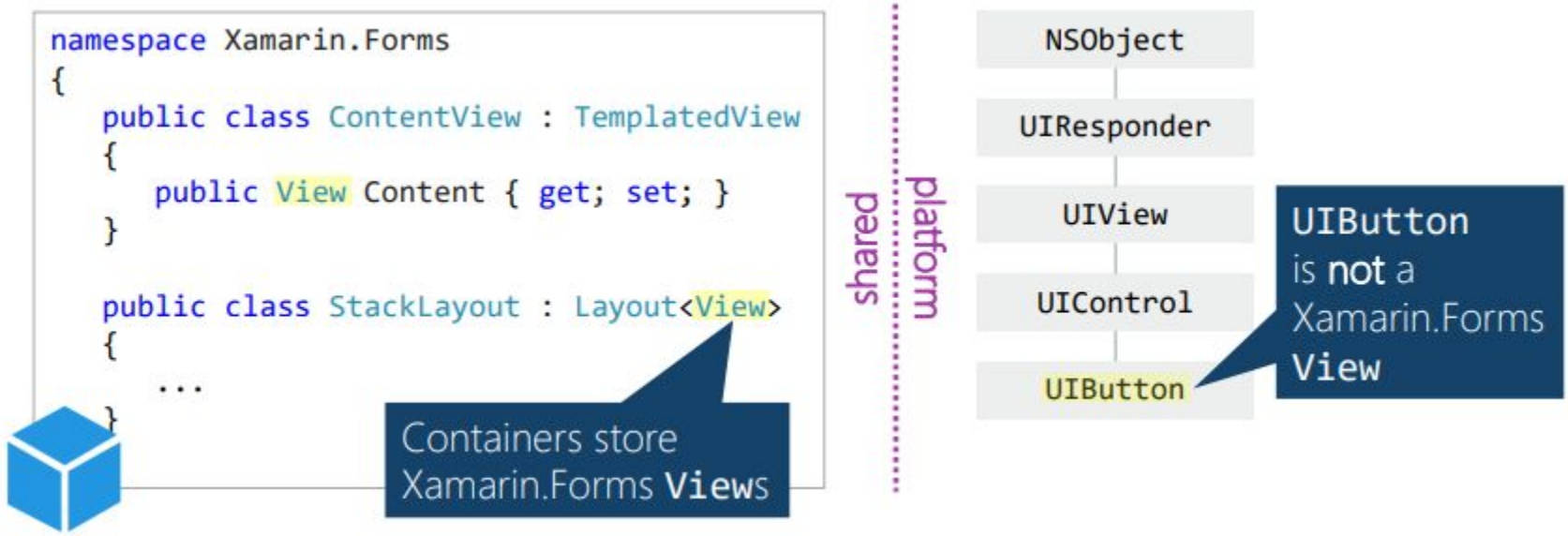
Controles Nativos en Xamarin.Forms

- Xamarin.Forms permite utilizar controles nativos embebidos directamente en la UI de Xamarin.Forms.



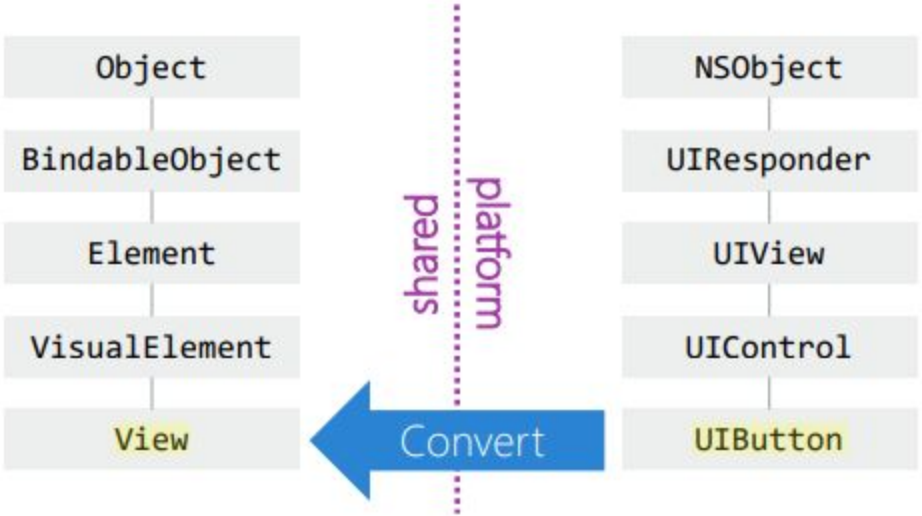
Incompatibilidad de Tipos

- Los contenedores de Xamarin.Forms derivan de **Xamarin.Forms.View**, por lo que controles nativos son incompatibles.

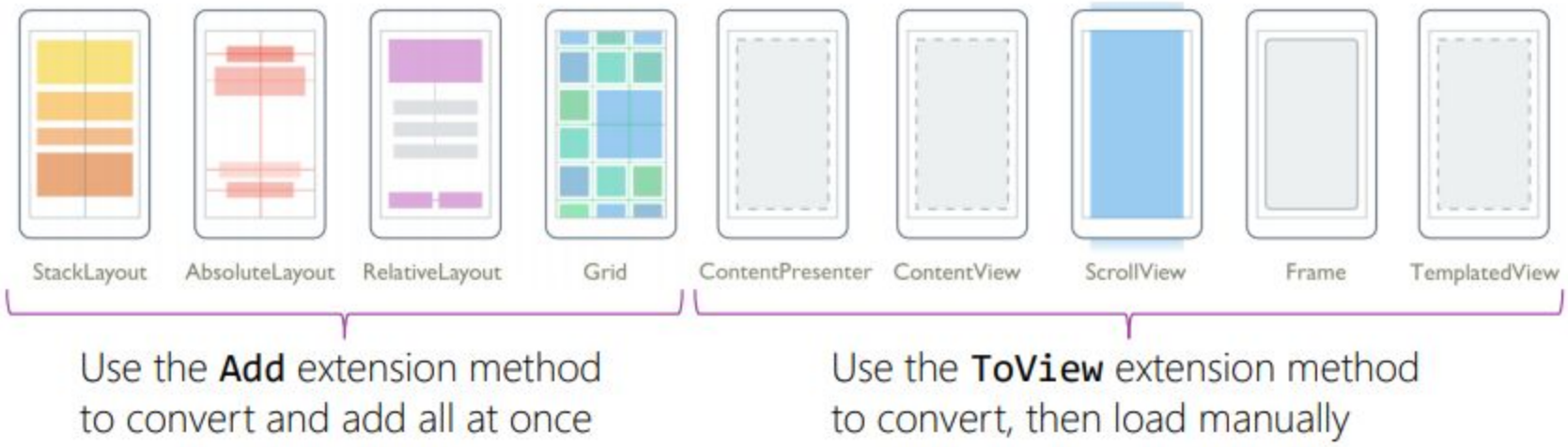


Conversión de Tipos

- Debido a la incompatibilidad, los controles nativos deben ser convertidos a **Xamarin.Forms.View** antes de que puedan ser añadidos a cualquier contenedor de layout de Xamarin.Forms (Grid,StackLayout,etc).



¿Cómo Añadir Controles Nativos?



Métodos de Extensión

- Xamarin.Forms provee métodos de extensión en cada plataforma para agregar controles nativos a cualquier layout de Xamarin.Forms

```
namespace Xamarin.Forms.Platform.Android
{
  public static class LayoutExtensions
  {
    public static
      void Add(this IList<Xamarin.Forms.View> children, Android.Views.View view, ...)
      {
        ...
      }
  }
}
```

The methods extend **IList** because that is the type of a layout's **Children** collection

Add an Android **View** to a collection of Xamarin.Forms **Views**

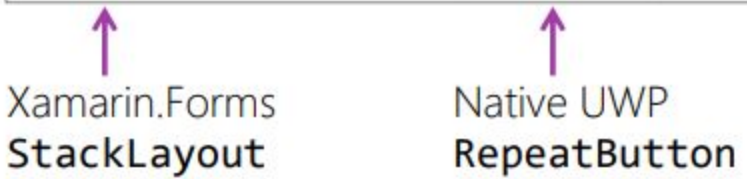
Añadiendo Control Nativo a Layout

- El método **Add** permite añadir controles nativos a layouts que posean una colección **Children.**

```
var xfStack = new Xamarin.Forms.StackLayout();

var uwpButton = new Windows.UI.Xaml.Controls.Primitives.RepeatButton();

xfStack.Children.Add(uwpButton);
```



Conversión de Controles

- Xamarin.Forms provee un método llamado **ToView** que convierte un control nativo en **Xamarin.Forms.View.**

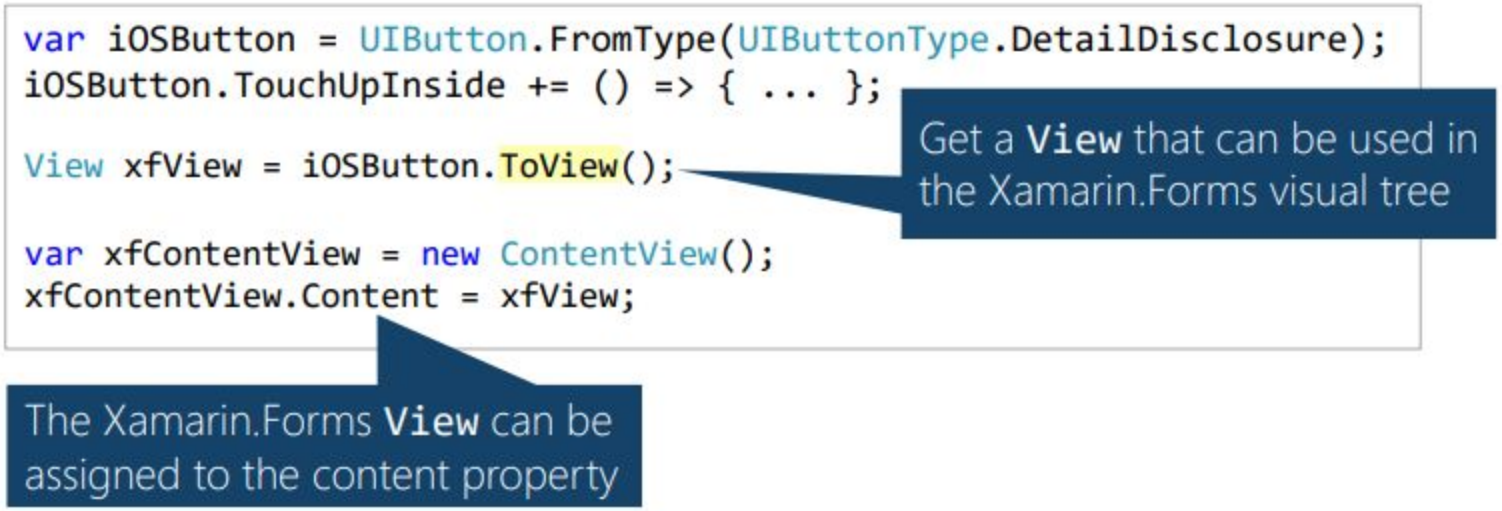
```
namespace Xamarin.Forms.Platform.Android
{
  public static class LayoutExtensions
  {
    public static Xamarin.Forms.View ToView(this Android.Views.View view, ...)
    {
      ...
    }
  }
}
```

Return a Xamarin.Forms **View**

Defined on each platform's native base visual type

Añadiendo Controles Convertidos

- Después de realizar la conversión del control nativo utilizando el método **ToView**, se agrega el resultado a un contenedor de layout.



Controles Embebidos en PCLs

- Al colocar la UI en un PCL, los controles nativos deben de ser agregados desde el proyecto nativo a través de una abstracción.

PCL defines the interface

```
public interface ICheckBoxFactory
{
    Xamarin.Forms.View GetCheckBox(string title, Action Checked);
}
```

Each native project implements it

```
class CheckBoxFactory : ICheckBoxFactory
{
    Context context;
    public Xamarin.Forms.View GetCheckBox(string title, Action Checked)
    {
        var cb = new Android.Widget.CheckBox(context) { Text = title };
        cb.CheckedChange += (s, e) => Checked();
        return cb.ToView();
    }
}
```

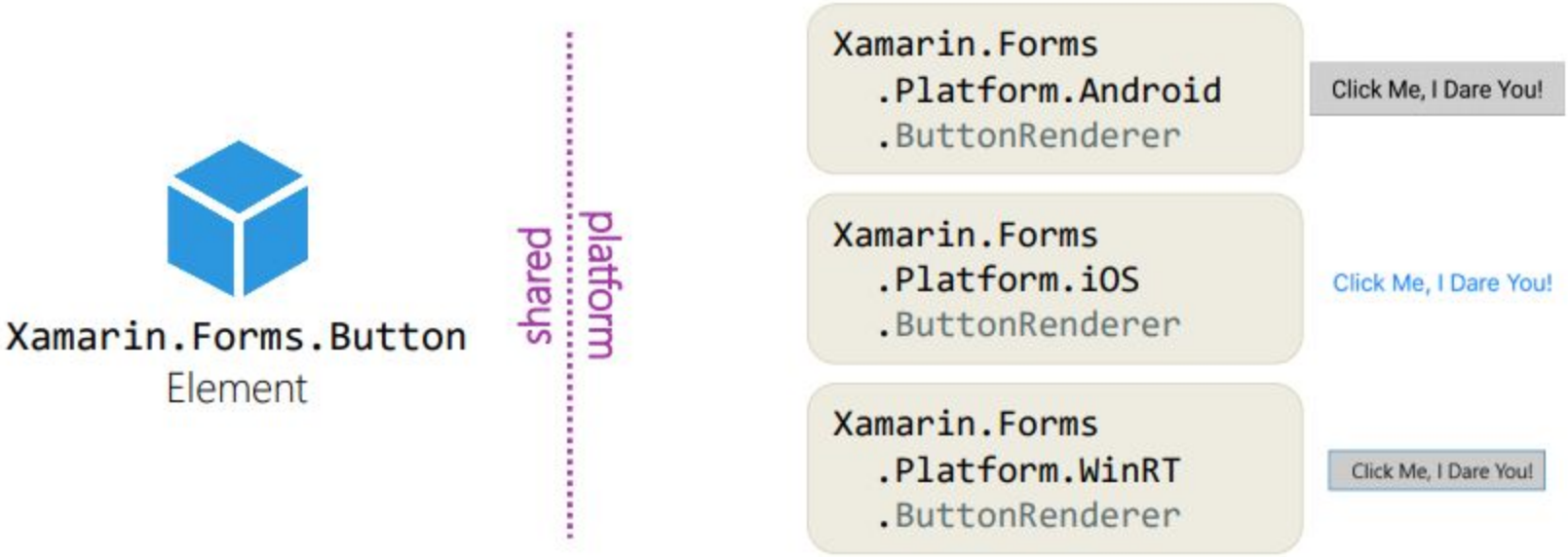
Use **ToView** to return a Xamarin Forms view

Demostración

Agregando un control nativo a layout de Xamarin.Forms

¿Que es un Renderizador?

- Código que traduce un elemento de Xamarin.Forms a un control específico de alguna plataforma en particular.



Renderizadores por Defecto

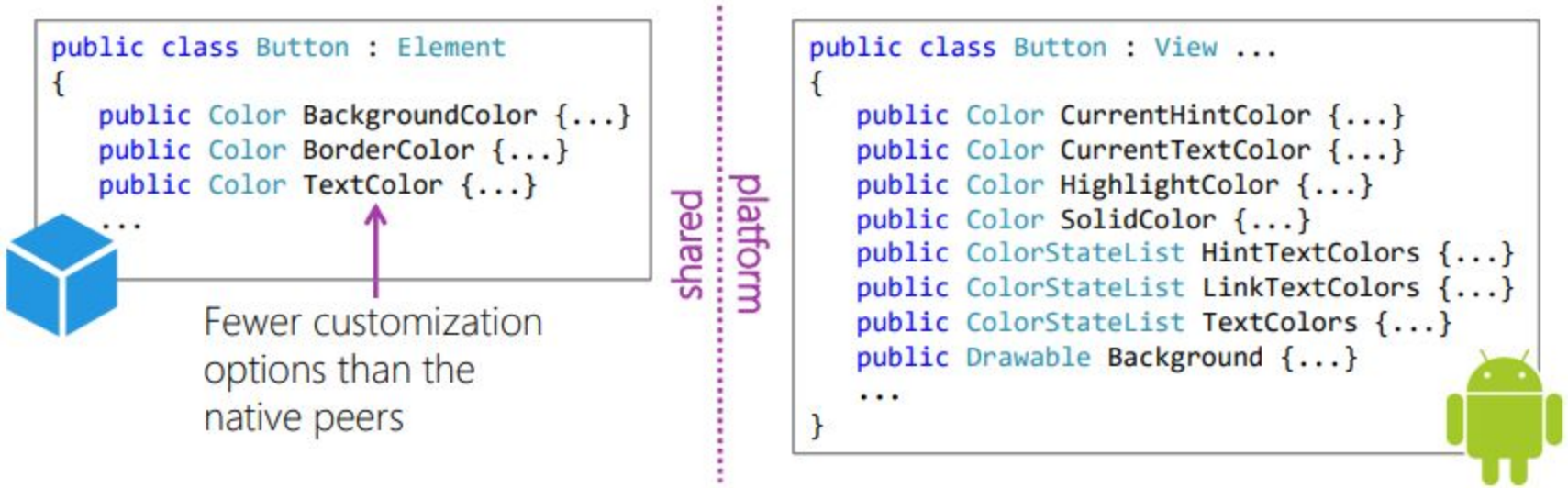
- Xamarin.Forms provee renderizadores para cada elemento

XF Element	Button	ContentPage	ContentView	EntryCell	…
iOS	ButtonRenderer	PageRenderer	ViewRenderer	EntryCellRenderer	…
Android	ButtonRenderer	PageRenderer	ViewRenderer	EntryCellRenderer	…
Windows	ButtonRenderer	PageRenderer	ViewRenderer	EntryCellRenderer	…

A complete list of renderers for each platform is available here:
https://developer.xamarin.com/guides/xamarin-forms/custom-renderer/renderers/

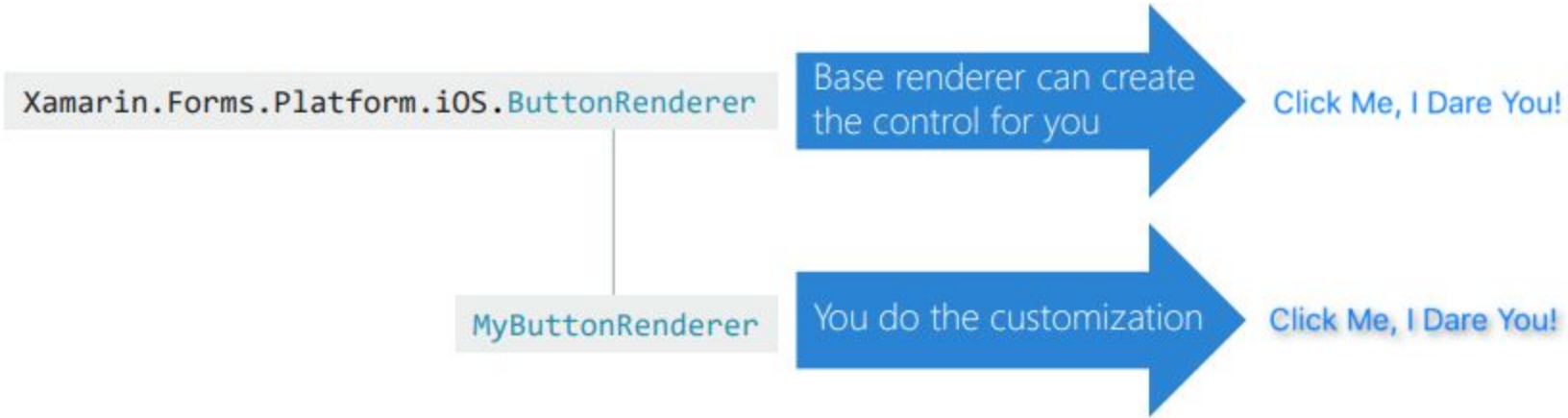
¿Porque personalizar un Renderizador?

- Las API's de Xamarin.Forms son limitadas en cuanto a cambiar la apariencia y comportamiento de los controles refiere, y los renderizadores personalizados permiten acceder a **TODAS** las propiedades nativas.



Tareas de un Renderizador Personalizado

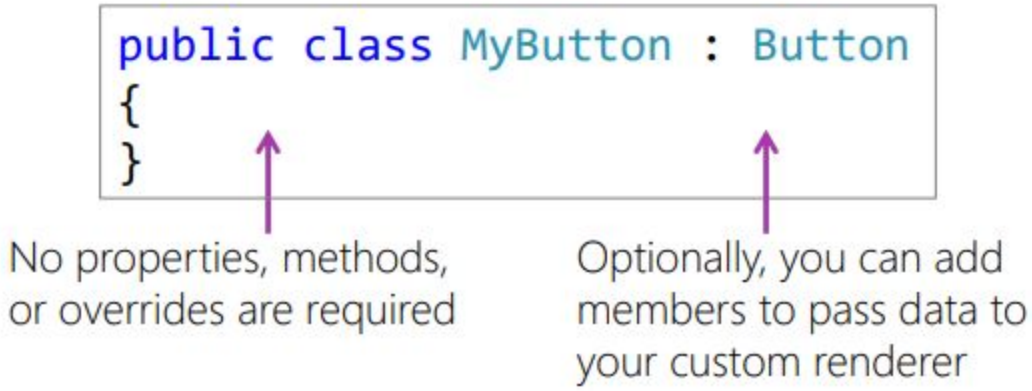
- Tienen dos tareas: Crear el control nativo y después personalizarlo haciendo uso de las API’s nativas de cada plataforma.



Personalizando un Renderizador

1. Instanciar el Elemento

- Crear una clase derivada del elemento a personalizar en el PCl.



2. Instanciar el Renderizador

- Crear una clase derivada del renderizador para el control nativo deseado en cada plataforma.

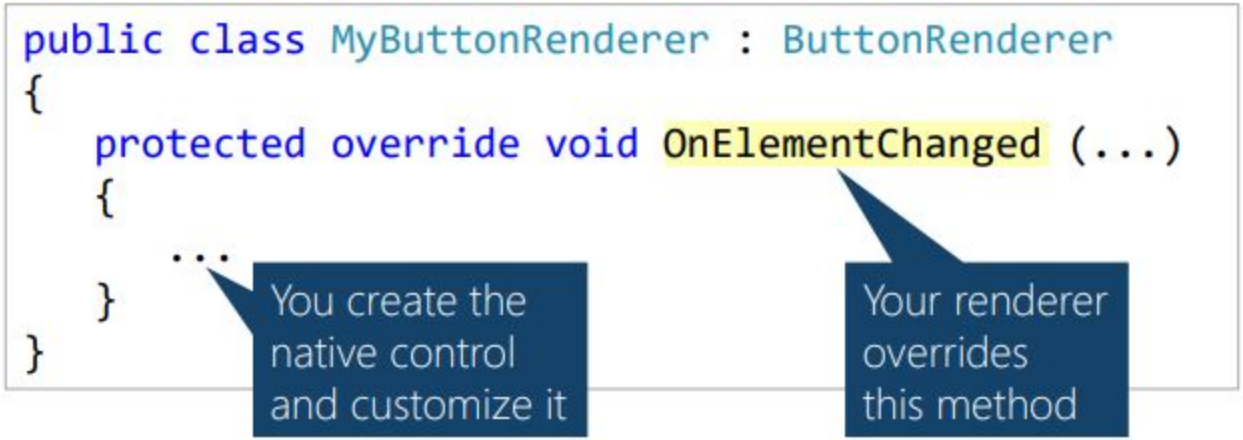
```
public class MyButtonRenderer : Xamarin.Forms.Platform.Android.ButtonRenderer
{
    ...
}
```

```
public class MyButtonRenderer : Xamarin.Forms.Platform.iOS.ButtonRenderer
{
    ...
}
```

```
public class MyButtonRenderer : Xamarin.Forms.Platform.UWP.ButtonRenderer
{
    ...
}
```

Ciclo de Vida de un Renderizador

- El método **OnElementChanged** es llamado cuando el renderizador recibe el elemento de Xamarin.Forms, y es aquí donde se hace la personalización del control antes de presentarlo al usuario.



Crear el Control Nativo

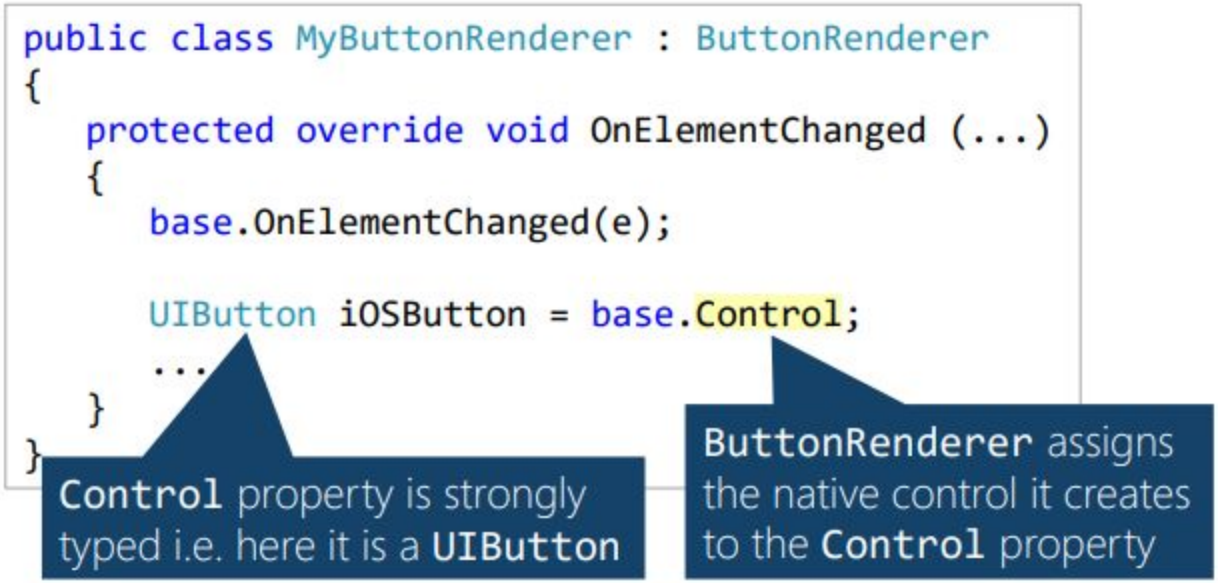
- El método **base.OnElementChanged** se encarga de crear el control nativo.

```
public class MyButtonRenderer : ButtonRenderer
{
    protected override void OnElementChanged (...)
    {
        base.OnElementChanged(e);
        ...
    }
}
```

ButtonRenderer creates the native control

Accesando al Control Nativo

- El acceso al control nativo es proporcionado a través de la propiedad **Control.**



Personalizando un Control Nativo

- Se utilizan las APIs nativas sobre la propiedad **Control** para llevar a cabo la personalización del mismo.

```
public class MyButtonRenderer : ButtonRenderer
{
    protected override void OnElementChanged (...)
    {  ...
        base.Control.Layer.ShadowOpacity = 1.0f;
    }
}
```

Use native APIs

3. Exportando un Renderizador

- Para conectar el elemento de Xamarin.Forms al renderizador personalizado del lado del proyecto nativo se utiliza una directiva de ensamblado.

```
[assembly: ExportRenderer(typeof(MyButton), typeof(MyButtonRenderer))]
```

The Xamarin.Forms element type

The platform-specific renderer visualizing the element

Consumiendo el Renderizador Personalizado

- Para consumirlo desde el PCL, lo único que se debe hacer es crear una instancia ya sea en código/XAML y utilizarlo normalmente.

Can use C#

```
var button = new MyButton();
myGrid.Children.Add(button);
```

Can use XAML

```
<ContentPage xmlns:local="clr-namespace:MyApp" ... />
   <local:MyButton Text="Click Me, I Dare You!" />
</ContentPage>
```

Demostración

Personalizando un renderizador

Gracias! :)

Referencias:

MSDN
WintellectNOW

Microsoft
CERTIFIED
Professional