Xamarin, Desarrollo rápido de aplicaciones móviles.

[4.1 Custom Render Xamarin.Forms 2](#_Toc523084936)

[4.1.1 Ejemplo 2](#_Toc523084937)

[4.2 Dependency Services 2](#_Toc523084938)

[4.2.1 Ejemplo 3](#_Toc523084939)

# 4.1 Custom Render Xamarin.Forms

Las interfases de usuario de Xamarin.Forms son presentada mediante los controles nativos de cada plataforma, lo que permite a las aplicaciones conserven un comportamiento y apariencia de cada plataforma.

Los custom renderers permite al desarrollador reemplazar este control para personalizarlo a la apariencia y comportamiento de acuerdo con lo necesitado.

A screenshot of a cell phone screen with text

Description generated with high confidence

Todos los controles de Xamarin.Forms tienen un “representador” que lo acompaña para cada plataforma que crea una instancia del control nativo. En el siguiente link, podras encontrar una lista de las clases de control nativo con su implementación en xamarin.Forms

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/custom-renderer/renderers>

## 4.1.1 Ejemplo

**Control**

1. **namespace** CustomRenderer
2. {
3. **public** **class** MyEntry : Entry
4. {
5. }
6. }

**Customización por plataforma**

1. /\*----- Android -----\*/
3. **using** Xamarin.Forms.Platform.Android;
4. **using** Xamarin.Forms;
5. **using** CustomRenderer;
6. **using** CustomRenderer.Android;
7. **using** Android.Content;
9. [assembly: ExportRenderer(**typeof**(MyEntry), **typeof**(MyEntryRenderer))]
10. **namespace** CustomRenderer.Android
11. {
12. **class** MyEntryRenderer : EntryRenderer
13. {
14. **public** MyEntryRenderer(Context context) : **base**(context)
15. {
16. }
18. **protected** **override** **void** OnElementChanged(ElementChangedEventArgs<Entry> e)
19. {
20. **base**.OnElementChanged(e);
22. **if** (Control != **null**)
23. {
24. Control.SetBackgroundColor(global::Android.Graphics.Color.LightGreen);
25. }
26. }
27. }
28. }
30. /\*----- IOS -----\*/
32. **using** Xamarin.Forms.Platform.iOS;
33. **using** Xamarin.Forms;
34. **using** UIKit;
35. **using** CustomRenderer;
36. **using** CustomRenderer.iOS;
38. [assembly: ExportRenderer (**typeof**(MyEntry), **typeof**(MyEntryRenderer))]
39. **namespace** CustomRenderer.iOS
40. {
41. **public** **class** MyEntryRenderer : EntryRenderer
42. {
43. **protected** **override** **void** OnElementChanged (ElementChangedEventArgs<Entry> e)
44. {
45. **base**.OnElementChanged (e);
47. **if** (Control != **null**) {
48. // do whatever you want to the UITextField here!
49. Control.BackgroundColor = UIColor.FromRGB (204, 153, 255);
50. Control.BorderStyle = UITextBorderStyle.Line;
51. }
52. }
53. }
54. }

**Implementación en Xaml**

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. **<ContentPage**
3. xmlns="http://xamarin.com/schemas/2014/forms"
4. xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5. **xmlns:local="clr-namespace:CustomRenderer;assembly=CustomRenderer"**
6. x:Class="CustomRenderer.MainPageXaml"**>**
7. **<StackLayout** VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"**>**
8. **<Label** Text="Hello, Custom Renderer!" **/>**
9. **<local:MyEntry Text="In Shared Code" />**
10. **</StackLayout>**
11. **</ContentPage>**

# 4.2 Dependency Services

Un dependency service permite que las aplicaciones puedan hacer llamadas a funciones especificas de cada plataforma desde el código compartido. Esto le permite a una aplicación de xamarin.forms hacer, lo que cualquier aplicación nativa podría hacer.

Las aplicaciones Xamarin.Forms necesitan 4 componentes para poder utilizar un DependencyService:

* Interfaz: Se define la funcionalidad requerida.
* Implementación por plataforma: Las clases que implementan la interfaz deben agregarse a cada proyecto de la plataforma.
* Registro: Cada clase de implementación debe registrarse con DependencyService, para ser habilitatado.
* La llamada al DependencyService: Solicita de la implementación de la interfaz.

A screenshot of a cell phone

Description generated with very high confidence

## 4.2.1 Ejemplo

**Interfaz**

1. **public** **interface** ITextToSpeech {
2. **void** Speak ( **string** text ); //note that interface members are public by defaul
3. }

**Implementación por plataforma**

1. /\*-------------IOS -------------- \*/
3. **namespace** UsingDependencyService.iOS
4. {
5. **public** **class** TextToSpeech\_iOS : ITextToSpeech
6. {
7. **public** **void** Speak (**string** text)
8. {
9. var speechSynthesizer = **new** AVSpeechSynthesizer ();
11. var speechUtterance = **new** AVSpeechUtterance (text) {
12. Rate = AVSpeechUtterance.MaximumSpeechRate/4,
13. Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
14. Volume = 0.5f,
15. PitchMultiplier = 1.0f
16. };
18. speechSynthesizer.SpeakUtterance (speechUtterance);
19. }
20. }
21. }
23. /\*----------- Android ------\*/
25. **namespace** UsingDependencyService.Android
26. {
27. **public** **class** TextToSpeech\_Android : Java.Lang.Object, ITextToSpeech, TextToSpeech.IOnInitListener
28. {
29. TextToSpeech speaker;
30. **string** toSpeak;
32. **public** **void** Speak(**string** text)
33. {
34. toSpeak = text;
35. **if** (speaker == **null**)
36. {
37. speaker = **new** TextToSpeech(MainActivity.Instance, **this**);
38. }
39. **else**
40. {
41. speaker.Speak(toSpeak, QueueMode.Flush, **null**, **null**);
42. Debug.WriteLine("spoke " + toSpeak);
43. }
44. }
46. #region IOnInitListener implementation
47. **public** **void** OnInit(OperationResult status)
48. {
49. **if** (status.Equals(OperationResult.Success))
50. {
51. Debug.WriteLine("speaker init");
52. speaker.Speak(toSpeak, QueueMode.Flush, **null**, **null**);
53. }
54. **else**
55. {
56. Debug.WriteLine("was quiet");
57. }
58. }
59. #endregion
60. }
61. }

**Registro**

1. /\*-------------IOS -------------- \*/
3. [assembly: Dependency(**typeof**(TextToSpeech\_iOS))]
5. /\*----------- Android ------\*/
7. [assembly: Dependency(**typeof**(TextToSpeech\_Android))]

**Llamada a Dependency Services**

1. DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");