Divisas MVVM - #NoCodeBehind

1. Cree un proyecto Xamarin Forms, llamado Divisas y asegúrese que el “**Welcome to Xamarin forms**” le esté funcionando en las 3 plataformas:

|  |  |  |
| --- | --- | --- |
| Android  C:\Users\Usuario\Desktop\Nexus 5 (Lollipop) Screenshot 1.png | iOS  C:\Users\Usuario\Downloads\Simulator Screen Shot 2.02.2017, 1.40.05 p.m..png | UWP |

1. Copia todo el código del **App.cs**, elimina el archivo y créalo nuevamente utilizando la plantilla de **XAML Page**. Debe de quedar de esta manera:

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MVVM.App">

</Application>

Ahora vamos al **code behind** y vamos reemplazar por el código que tenemos en la porta papeles, colocando el **partial class** y el **InitializeComponent**(), debe quedar así:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Xamarin.Forms;

namespace MVVM

{

public partial class App : Application

{

public App()

{

InitializeComponent();

// The root page of your application

MainPage = new ContentPage

{

Content = new StackLayout

{

VerticalOptions = LayoutOptions.Center,

Children = {

new Label {

HorizontalTextAlignment = TextAlignment.Center,

Text = "Welcome to Xamarin Forms!"

}

}

}

};

}

protected override void OnStart()

{

// Handle when your app starts

}

protected override void OnSleep()

{

// Handle when your app sleeps

}

protected override void OnResume()

{

// Handle when your app resumes

}

}

}

1. Vamos a modificar nuestra App para crear el diccionario de recursos, inicialmente con los colores que va a manejar nuestra aplicación, debe quedar así:

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DivisasMVVM.App">

<Application.Resources>

<ResourceDictionary>

</ResourceDictionary>

</Application.Resources>

</Application>

1. Vamos a crear la carpeta **Pages** y dentro de esta vamos a crear el **MainPage**:

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DivisasMVVM.Pages.MainPage"

Title="Divisas">

<Label Text="Main Page" VerticalOptions="Center" HorizontalOptions="Center" />

</ContentPage>

1. Cambiamos el **App.xaml.cs** por:

public App()

{

InitializeComponent();

MainPage = new NavigationPage(new MainPage());

}

Y probamos que nuestro proyecto aun funcione:

|  |  |  |
| --- | --- | --- |
| Android  C:\Users\Usuario\Desktop\Nexus 5 (Lollipop) Screenshot 2.png | iOS  C:\Users\Usuario\Downloads\Simulator Screen Shot 2.02.2017, 2.02.31 p.m..png | UWP |

1. Creamos la carpeta **ViewModels** y dentro de esta vamos a crear el **MainViewModel**, por el momento vacía:

public class MainViewModel

{

}

1. Crearemos esta clase para poder implementar el patrón **Locator**. Para tal fin vamos a crear la carpeta **Infrastructure** y dentro de esta la clase **InstanceLocator**

public class InstanceLocator

{

public MainViewModel Main { get; set; }

public InstanceLocator()

{

Main = new MainViewModel();

}

}

1. Vamos hacer que nuestro **InstanceLocator** sea un recurso general para toda la aplicación. Vamos al **App.xaml** y le hacemos las siguientes modificaciones:

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:infra="clr-namespace:DivisasMVVM.Infrastructure;assembly=DivisasMVVM"

x:Class="DivisasMVVM.App">

<Application.Resources>

<ResourceDictionary>

<!-- Locator -->

<infra:InstanceLocator x:Key="Locator"></infra:InstanceLocator>

</ResourceDictionary>

</Application.Resources>

</Application>

1. Ahora hagamos que nuestra **MainPage**, use el **MainViewModel**, y modificamos esta página para que muestre nuestra página como la necesitamos:

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DivisasMVVM.Pages.MainPage"

Title="Divisas"

BindingContext="{Binding Main, Source={StaticResource Locator}}">

<ScrollView>

<StackLayout

Padding="8">

<Label

Text="Pesos:">

</Label>

<Entry

Keyboard="Numeric"

Placeholder="Entre valor en pesos...">

</Entry>

<Button

Text="Convertir">

</Button>

<Label

Text="Dólares:">

</Label>

<Entry

IsEnabled="false"

Placeholder="Valor en dólares...">

</Entry>

<Label

Text="Libras:">

</Label>

<Entry

IsEnabled="false"

Placeholder="Valor en libras...">

</Entry>

<Label

Text="Euros:">

</Label>

<Entry

IsEnabled="false"

Placeholder="Valor en euros...">

</Entry>

</StackLayout>

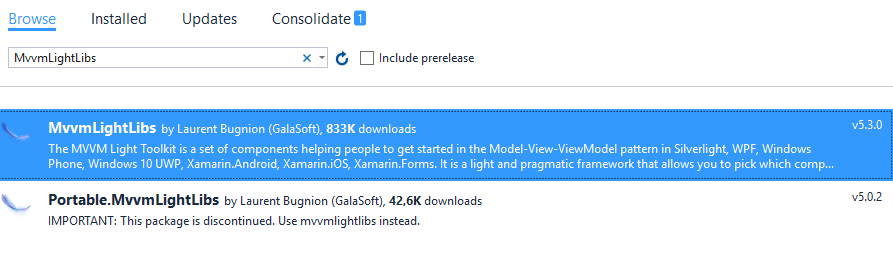
</ScrollView>

</ContentPage>

1. Probemos como queda nuestra interfaz de usuario:

|  |  |  |
| --- | --- | --- |
| Android | iOS  C:\Users\Usuario\Downloads\Simulator Screen Shot 2.02.2017, 3.14.19 p.m..png | UWP |

1. Debemos de instalar un paquete de Nuget, **MvvmLightLibs** al proyecto compartido:



1. Por buenas prácticas creemos la carpeta **Services** y dentro de esta el **DialogService**, que vamos a usar más adelante:

public class DialogService

{

public async Task ShowMessage(string title, string message)

{

await App.Current.MainPage.DisplayAlert(title, message, "Aceptar");

}

public async Task<bool> ShowConfirm(string title, string message)

{

return await App.Current.MainPage.DisplayAlert(title, message, "Sí", "No");

}

}

1. Modifiquemos nuestro **MainPage** para que los controles se “binden” al **MainViewModel**:

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DivisasMVVM.Pages.MainPage"

Title="Divisas"

BindingContext="{Binding Main, Source={StaticResource Locator}}">

<ScrollView>

<StackLayout

Padding="8">

<Label

Text="Pesos:">

</Label>

<Entry

Text="{Binding Pesos}"

Keyboard="Numeric"

Placeholder="Entre valor en pesos...">

</Entry>

<Button

Command="{Binding ConvertCommand}"

BackgroundColor="Navy"

TextColor="White"

HeightRequest="40"

BorderRadius="20"

Text="Convertir">

</Button>

<Label

Text="Dólares:">

</Label>

<Entry

Text="{Binding Dollars, StringFormat='${0:N2}'}}"

IsEnabled="false"

Placeholder="Valor en dólares...">

</Entry>

<Label

Text="Libras:">

</Label>

<Entry

Text="{Binding Pounds, StringFormat=' 0:N2}'}"

IsEnabled="false"

Placeholder="Valor en libras...">

</Entry>

<Label

Text="Euros:">

</Label>

<Entry

Text="{Binding Euros, StringFormat='€{0:N2}'}"

IsEnabled="false"

Placeholder="Valor en euros...">

</Entry>

</StackLayout>

</ScrollView>

</ContentPage>

1. Y modifiquemos nuestro **MainViewModel**:

using DivisasMVVM.Services;

using GalaSoft.MvvmLight.Command;

using System.ComponentModel;

using System.Windows.Input;

namespace DivisasMVVM.ViewModels

{

public class MainViewModel : INotifyPropertyChanged

{

#region Attributes

private DialogService dialogService;

private decimal pesos;

private decimal dollars;

private decimal pounds;

private decimal euros;

#endregion

#region Properties

public decimal Pesos

{

set

{

if (pesos != value)

{

pesos = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Pesos"));

}

}

get

{

return pesos;

}

}

public decimal Dollars

{

set

{

if (dollars != value)

{

dollars = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Dollars"));

}

}

get

{

return dollars;

}

}

public decimal Pounds

{

set

{

if (pounds != value)

{

pounds = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Pounds"));

}

}

get

{

return pounds;

}

}

public decimal Euros

{

set

{

if (euros != value)

{

euros = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Euros"));

}

}

get

{

return euros;

}

}

#endregion

#region Constructors

public MainViewModel()

{

dialogService = new DialogService();

}

#endregion

#region Events

public event PropertyChangedEventHandler PropertyChanged;

#endregion

#region Commands

public ICommand ConvertCommand { get { return new RelayCommand(ConvertMoney); } }

private async void ConvertMoney()

{

if (Pesos <= 0)

{

await dialogService.ShowMessage("Error", "Debe ingresar un valor en pesos mayor a cero (0)");

return;

}

Dollars = Pesos / (decimal)2881.84438;

Pounds = Pesos / (decimal)3608.93372;

Euros = Pesos / (decimal)3101.61383;

}

#endregion

}

}

1. Probamos como queda nuestra aplicación terminada:

|  |  |  |
| --- | --- | --- |
| Android  C:\Users\Usuario\Desktop\Nexus 5 (Lollipop) Screenshot 1.png  C:\Users\Usuario\Desktop\Nexus 5 (Lollipop) Screenshot 2.png | iOS  C:\Users\Usuario\Downloads\Simulator Screen Shot 2.02.2017, 4.19.36 p.m..png  C:\Users\Usuario\Downloads\Simulator Screen Shot 2.02.2017, 4.19.50 p.m..png | UWP |