# XamarinMultiplataform

# App.xaml

<?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:MultiplataformApp.Infrastructure"

x:Class="MultiplataformApp.App">

<Application.Resources>

<!-- Application resource dictionary -->

<ResourceDictionary>

<!-- Locator -->

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

</ResourceDictionary>

</Application.Resources>

</Application>

# Folder Infrastructure/InstanceLocator.cs

namespace MultiplataformApp.Infrastructure

{

using MainViewModel;

public class InstanceLocator

{

public MainViewModel Main{ get; set; }

public InstanceLocator()

{

Main = new MainViewModel();

}

}

}

# Folder ViewModels/MainViewModels.cs

namespace MultiplataformApp.MainViewModel

{

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Net.Http;

using System.Windows.Input;

using GalaSoft.MvvmLight.Command;

using MultiplataformApp.Models;

using Newtonsoft.Json;

using Xamarin.Forms;

public class MainViewModel : INotifyPropertyChanged

{

#region Events

public event PropertyChangedEventHandler PropertyChanged;

#endregion

#region Attributes

bool \_isRunning;

bool \_isEnabled;

string \_result;

ObservableCollection<Rate> \_rates;

Rate \_sourceRate;

Rate \_targetRate;

#endregion

#region Propierties

public string Amount

{

get;

set;

}

public ObservableCollection<Rate> Rates

{

get

{

return \_rates;

}

set

{

if (\_rates != value)

{

\_rates = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(Rates)));

}

}

}

public Rate SourceRate

{

get

{

return \_sourceRate;

}

set

{

if (\_sourceRate != value)

{

\_sourceRate = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(SourceRate)));

}

}

}

public Rate TargetRate

{

get

{

return \_targetRate;

}

set

{

if (\_targetRate != value)

{

\_targetRate = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(TargetRate)));

}

}

}

public bool IsRunning

{

get{

return \_isRunning;

}

set{

if(\_isRunning != value)

{

\_isRunning = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(IsRunning)));

}

}

}

public bool IsEnabled

{

get

{

return \_isEnabled;

}

set

{

if (\_isEnabled != value)

{

\_isEnabled = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(IsEnabled)));

}

}

}

public string Result

{

get

{

return \_result;

}

set

{

if (\_result != value)

{

\_result = value;

PropertyChanged?.Invoke(

this,

new PropertyChangedEventArgs(nameof(Result)));

}

}

}

#endregion

#region Constructors

public MainViewModel()

{

LoadRates();

}

#endregion

#region Methods

async void LoadRates()

{

IsRunning = true;

Result = "Loading rates...";

try

{

var client = new HttpClient();

client.BaseAddress = new

Uri("http://apiexchangerates.azurewebsites.net");

var controller = "/api/Rates";

var response = await client.GetAsync(controller);

var result = await response.Content.ReadAsStringAsync();

if (!response.IsSuccessStatusCode)

{

IsRunning = false;

Result = result;

}

var rates = JsonConvert.DeserializeObject<List<Rate>>(result);

Rates = new ObservableCollection<Rate>(rates);

IsRunning = false;

IsEnabled = true;

Result = "Ready to convert";

}

catch (Exception ex)

{

IsRunning = false;

Result = ex.Message;

}

}

#endregion

#region Commands

public ICommand SwitchCommand

{

get

{

return new RelayCommand(Switch);

}

}

void Switch()

{

var aux = SourceRate;

SourceRate = TargetRate;

TargetRate = aux;

Convert();

}

public ICommand ConvertCommand

{

get

{

return new RelayCommand(Convert);

}

}

async void Convert()

{

if(string.IsNullOrEmpty(Amount))

{

await Application.Current.MainPage.DisplayAlert(

"Error",

"You must enter a value in amount.",

"Accept");

return;

}

decimal amount = 0;

if(!decimal.TryParse(Amount, out amount))

{

await Application.Current.MainPage.DisplayAlert(

"Error",

"You must enter a numeric value in amount.",

"Accept");

return;

}

if(SourceRate == null)

{

await Application.Current.MainPage.DisplayAlert(

"Error",

"You must select a source rate.",

"Accept");

return;

}

if (TargetRate == null)

{

await Application.Current.MainPage.DisplayAlert(

"Error",

"You must select a target rate.",

"Accept");

return;

}

var amountConverted = amount /

(decimal) SourceRate.TaxRate \*

(decimal) TargetRate.TaxRate;

Result = string.Format("{0} {1:C2} = {2} {3:C2}",

SourceRate.Code,

amount,

TargetRate.Code,

amountConverted);

}

#endregion

}

}

# Folder Views/MainView.xaml

<?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="MultiplataformApp.Views.MainView"

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

<ContentPage.Padding>

<OnPlatform

x:TypeArguments="Thickness"

iOS="20,30,20,10"

Android="10"/>

</ContentPage.Padding>

<ContentPage.Content>

<StackLayout>

<Label

FontAttributes="Bold"

FontSize="Large"

HorizontalOptions="Center"

Text="Foreign Exchange"

Margin="10">

</Label>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width=".5\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width=".3\*"/>

</Grid.ColumnDefinitions>

<Label

Grid.Column="0"

Grid.Row="0"

Text="Amount:"

VerticalOptions="Center">

</Label>

<Entry

Grid.Column="1"

Grid.Row="0"

Grid.ColumnSpan="2"

Text="{Binding Amount, Mode=TwoWay}"

Placeholder="Enter the amount to">

</Entry>

<Label

Grid.Column="0"

Grid.Row="1"

Text="Source rate:"

VerticalOptions="Center">

</Label>

<Picker

Grid.Column="1"

Grid.Row="1"

ItemDisplayBinding="{Binding Name}"

ItemsSource="{Binding Rates}"

SelectedItem="{Binding SourceRate}"

Title="Select a source rate...">

</Picker>

<Label

Grid.Column="0"

Grid.Row="2"

Text="Target rate:"

VerticalOptions="Center">

</Label>

<Picker

Grid.Column="1"

Grid.Row="2"

ItemDisplayBinding="{Binding Name}"

ItemsSource="{Binding Rates}"

SelectedItem="{Binding TargetRate}"

Title="Select a target rate...">

</Picker>

<Image

Grid.Column="2"

Grid.Row="1"

Grid.RowSpan="2"

HeightRequest="40"

Source="switchIcon.png"

WidthRequest="40">

<Image.GestureRecognizers>

<TapGestureRecognizer Command="{Binding SwitchCommand}"/>

</Image.GestureRecognizers>

</Image>

</Grid>

<ActivityIndicator

IsRunning="{Binding IsRunning, Mode=TwoWay}">

</ActivityIndicator>

<Button

Command="{Binding ConvertCommand}"

BackgroundColor="Navy"

BorderRadius="20"

HeightRequest="50"

FontAttributes="Bold"

IsEnabled= "{Binding IsEnabled, Mode=TwoWay}"

Text="Convert"

TextColor="White">

</Button>

<Label

BackgroundColor="Silver"

FontSize="Large"

HorizontalTextAlignment="Center"

Margin="0,10"

Text="{Binding Result, Mode=TwoWay}"

VerticalOptions="FillAndExpand"

VerticalTextAlignment="Center">

</Label>

</StackLayout>

</ContentPage.Content>

</ContentPage>