



Xamarin

Xamarin.Forms

Introduzione

Creare app iOS, Android, Mac, UWP, WPF con una singola UI codebase



#XamarinRocks

@cloudgen_verona



Francesco Bonacci

MSP – Xamarin Certified Developer

francesco.bonacci@studentpartner.com



@francedot



github.com/francedot



francesco-bonacci-70428a121

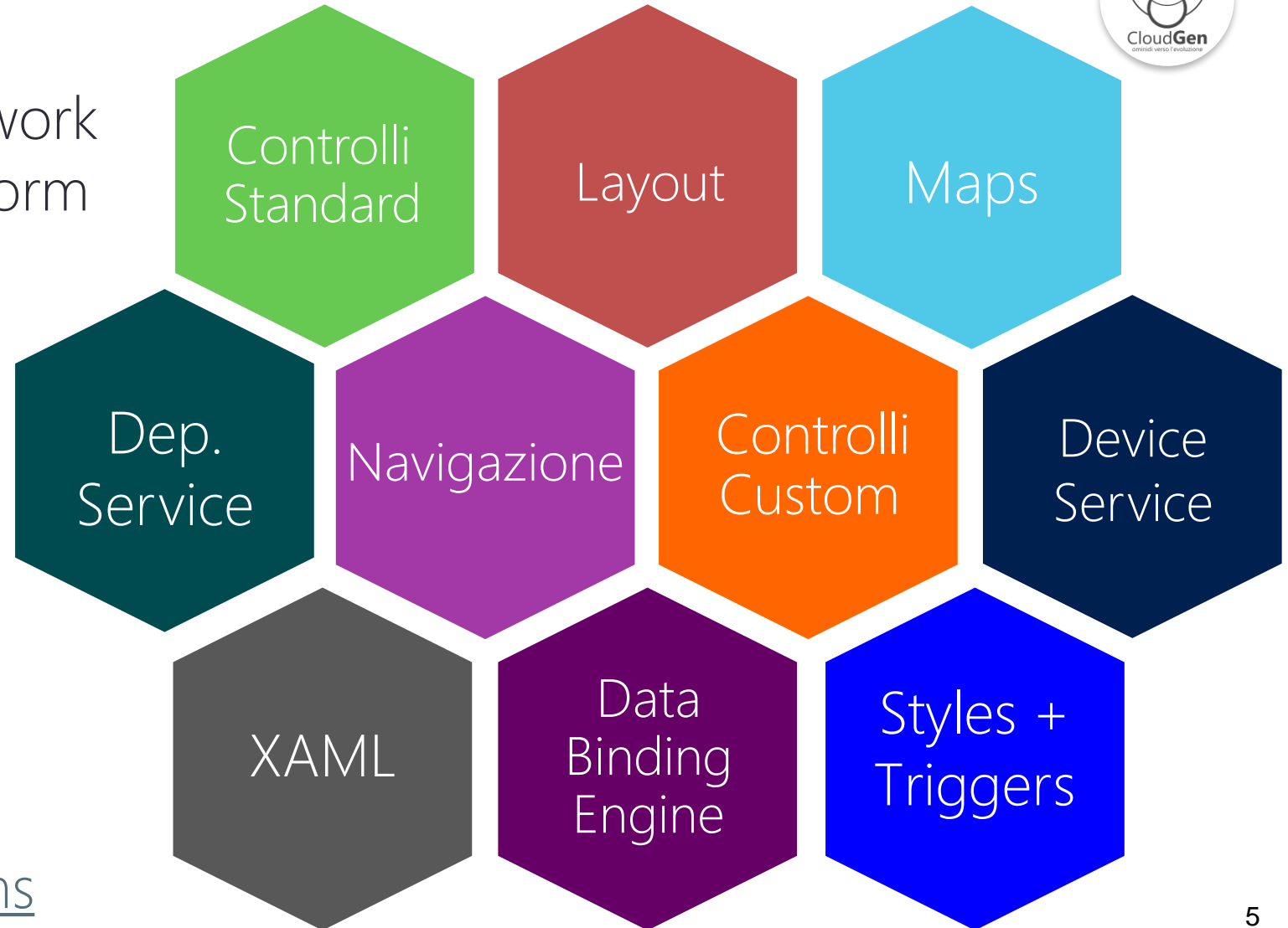


Xamarin.Forms è un framework per applicazioni cross-platform

Attualmente supportati:

- Android 4.0+
- iOS 6.1+
- ~~Windows Phone 8 (SL)~~
- ~~Windows 8.1 (SL/RT)~~
- UWP (Windows 10)
- Mac
- WPF - Merged PR

Next? Web! See [Ooui.Forms](#)

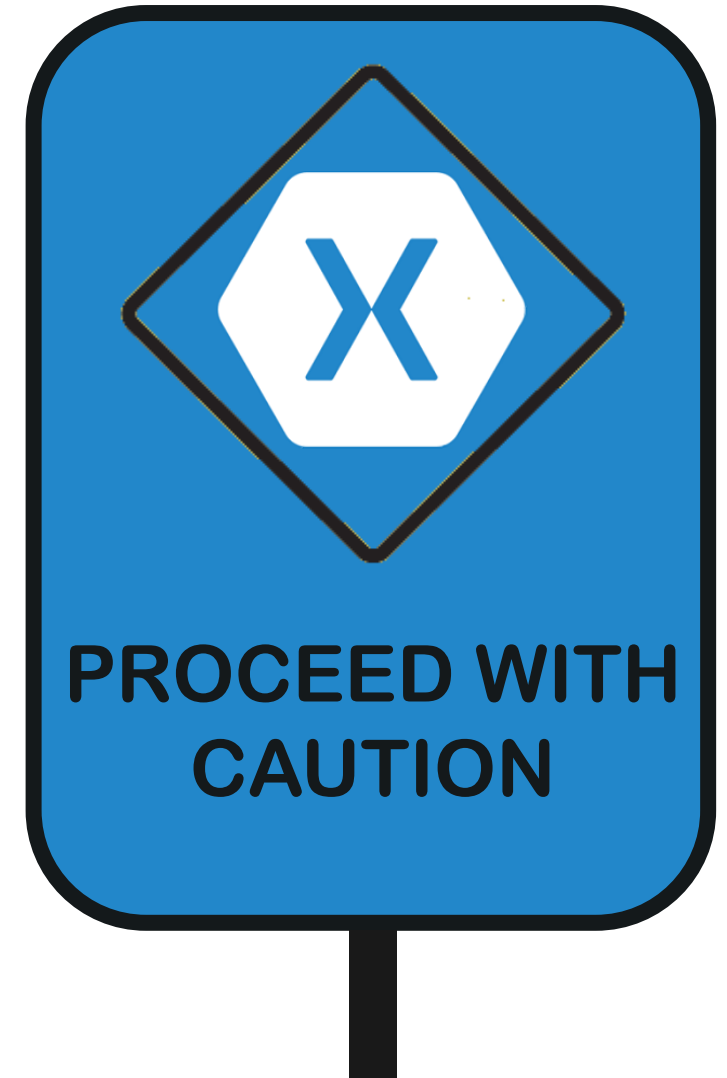


Xamarin.Forms - Quando?

Introduzione



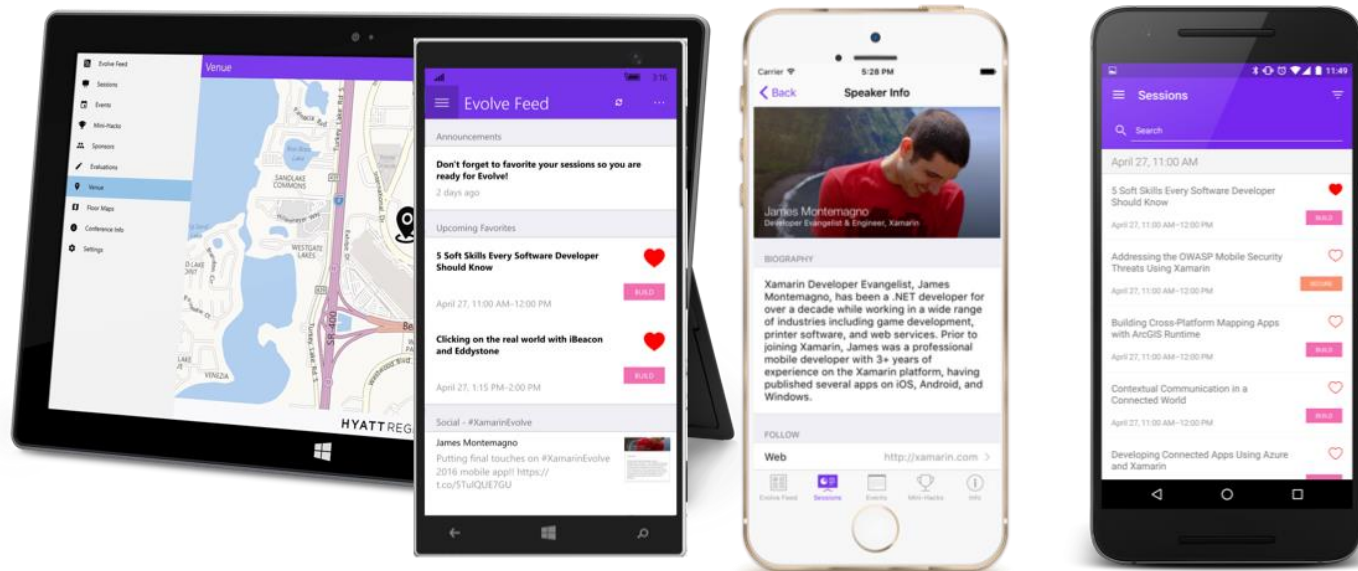
- Xamarin.Forms non è adatto a tutti i tipi di app
 - ✓ OK per applicazioni di utilità o data-entry
 - ✗ Non ideale se l'obiettivo finale è produrre effetti grafici sfavillanti
- Primo approccio per i WPF-guru
- Spesso usato da team per la prima fase di prototipaggio dell'applicazione → Poi eventualmente si passa a Native



Xamarin.Forms non è da sottovalutare:

- Si possono creare UI accattivanti e anche complesse

github.com/xamarin/hq/app-evolve





Xamarin.Forms non è da sottovalutare:

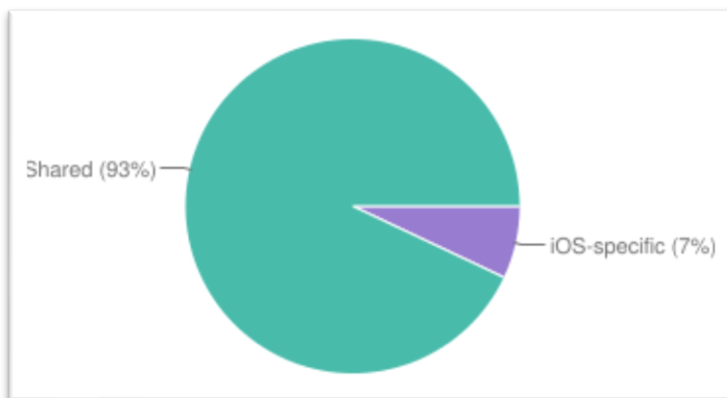
- Si possono creare UI accattivanti e anche complesse
- Si evita di replicare il codice della UI

Codice
cross-platform

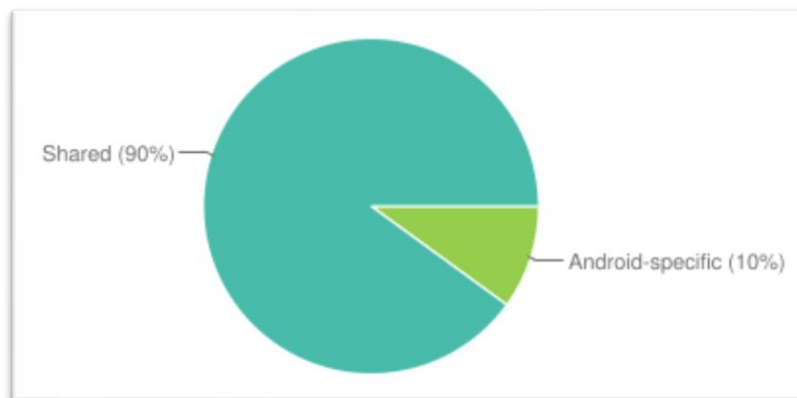
github.com/xamarin/hq/app-evolve

*Statistiche prese dall'app
Xamarin Evolve 2016

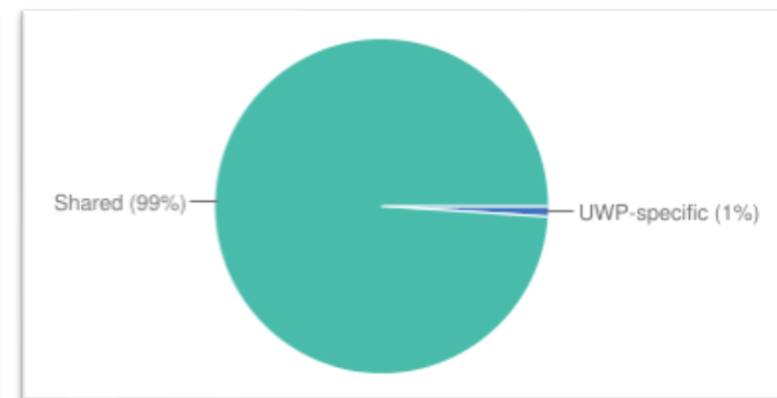
iOS



Android

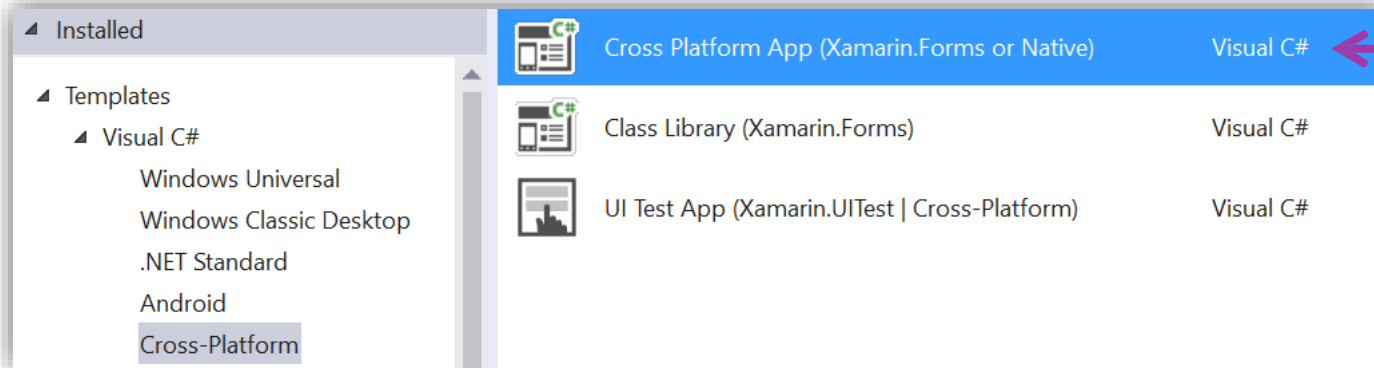


UWP



Creare un App Xamarin.Forms - Windows

Introduzione

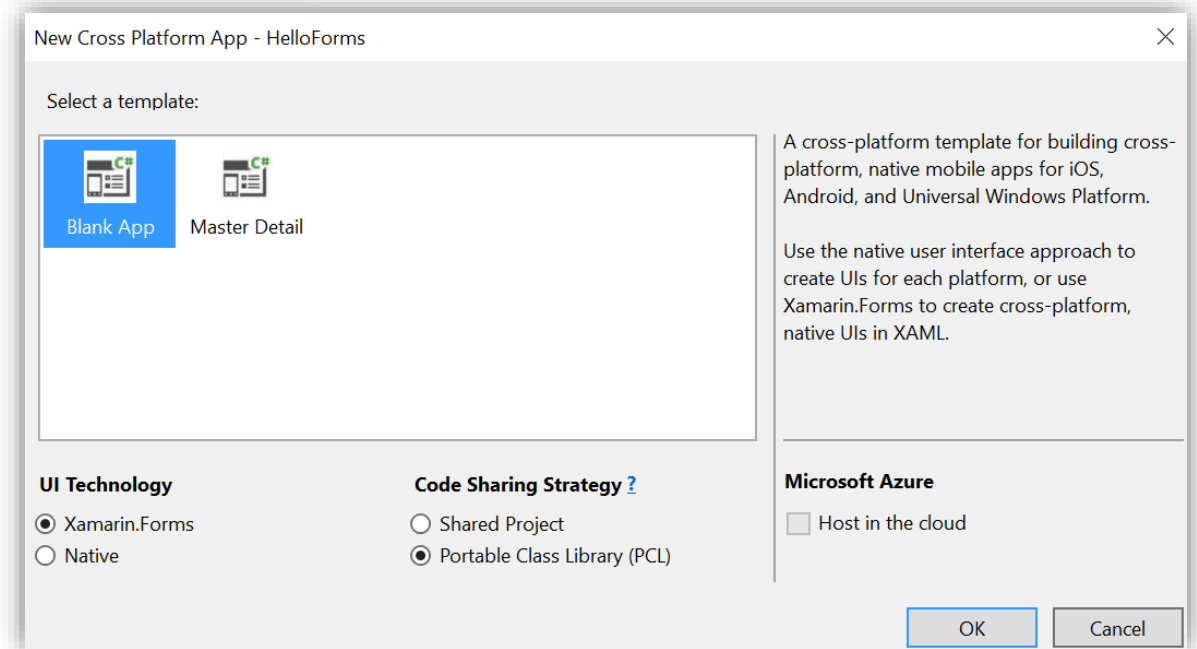


1. Scegliere Tipologia di progetto Visual C# → Cross-Platform → Cross Platform App

2. Selezionare il Template Blank App (Singola Pagina vuota)

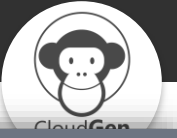


3. Come UI Technology indicare Xamarin.Forms

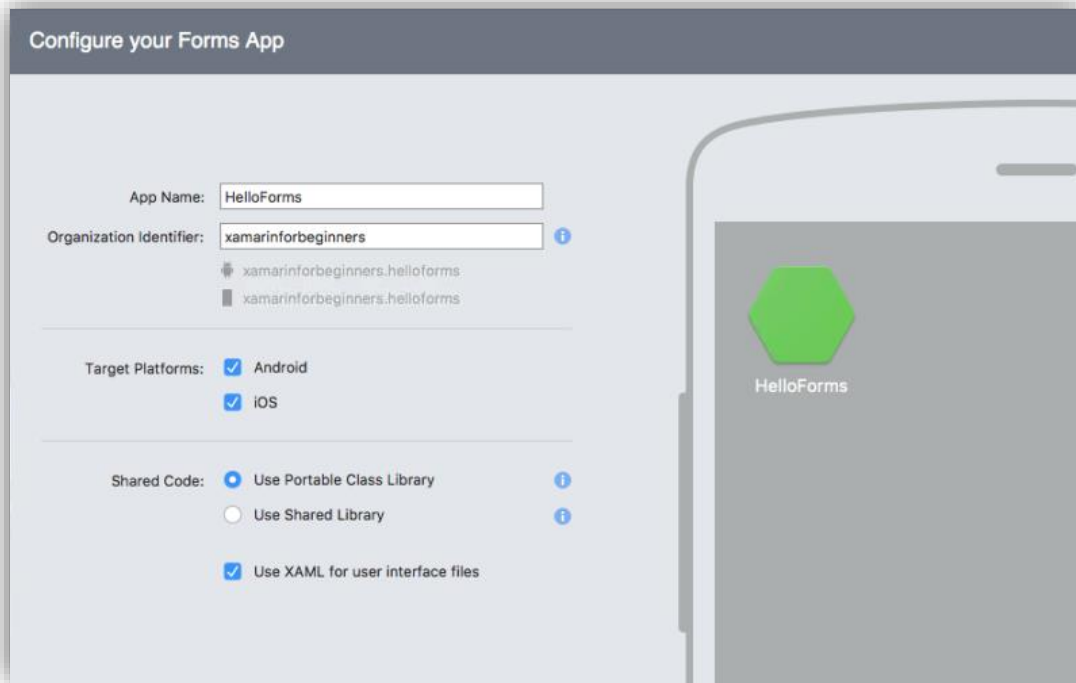
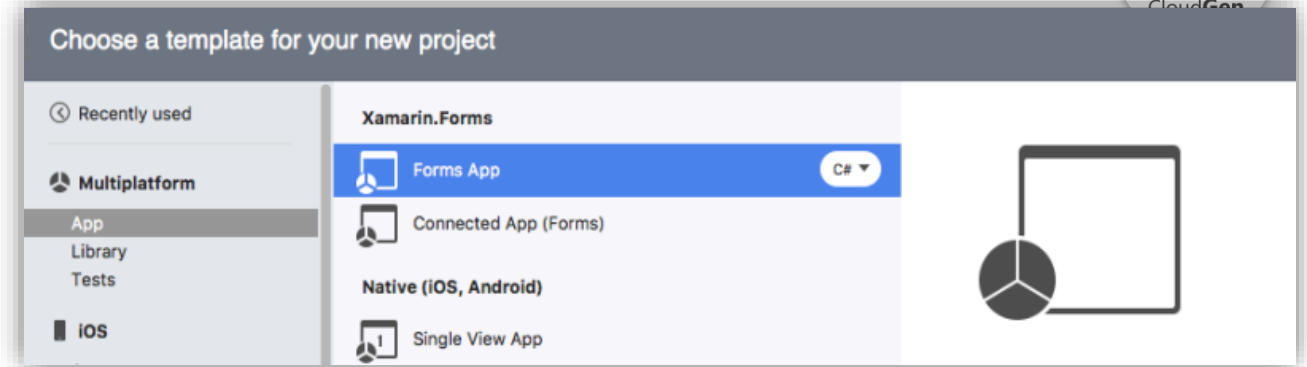


Creare un App Xamarin.Forms - Mac

Introduzione



1. Scegliere Tipologia di Progetto
Multiplatform → App →
Forms App (Blank App)



2. Configurare il progetto:
 - Selezionare le piattaforme target (iOS e Android)
 - Optare tra PCL/.NET Standard e SP
 - Scegliere se utilizzare markup Xaml per le classi parziali di App e Page

Struttura del Progetto

Introduzione

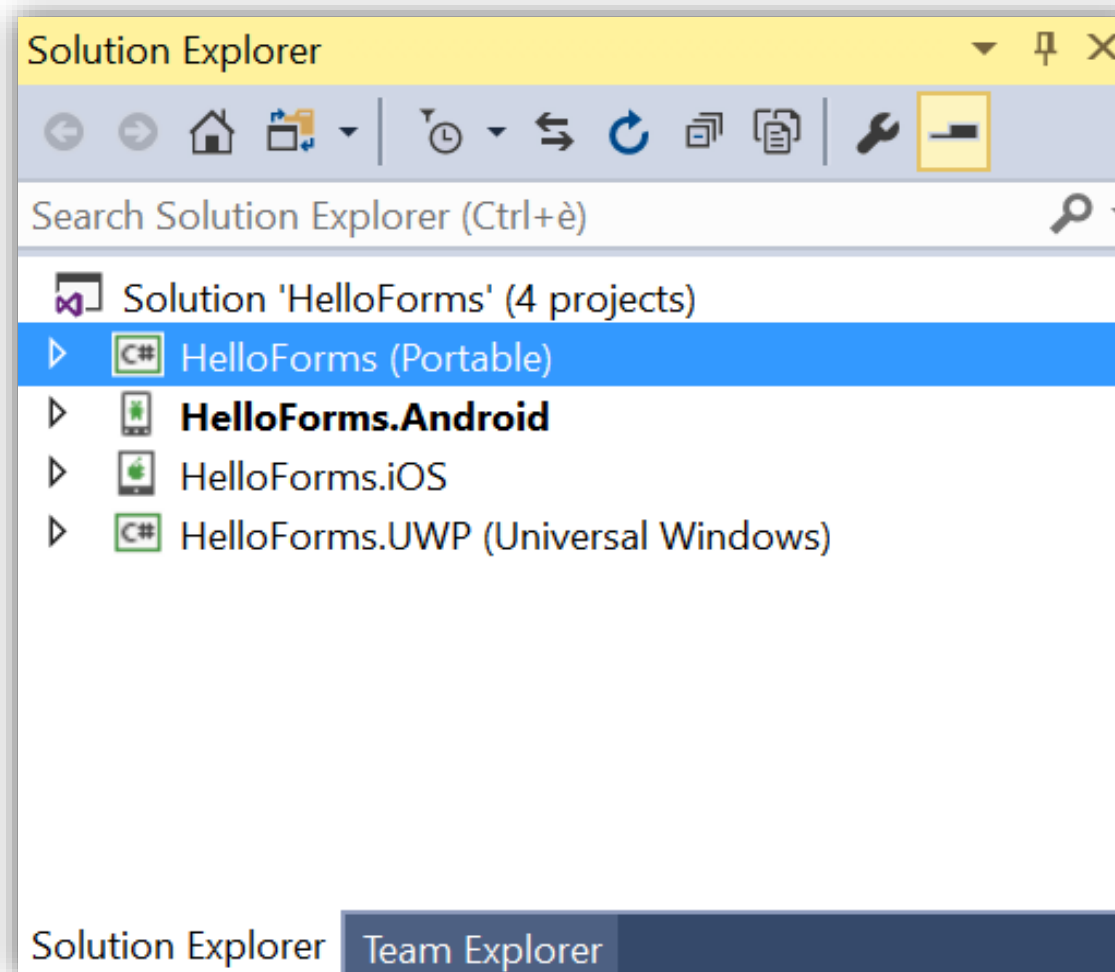


I template di App Xamarin.Forms creano 4 progetti

PCL/.NET Standard o SP →

Progetti
platform-specific

In Xamarin.Forms le PCL e SP sono utilizzate non solo per condividere modello e business logic, ma anche codice relativo alla UI (C# o Xaml)

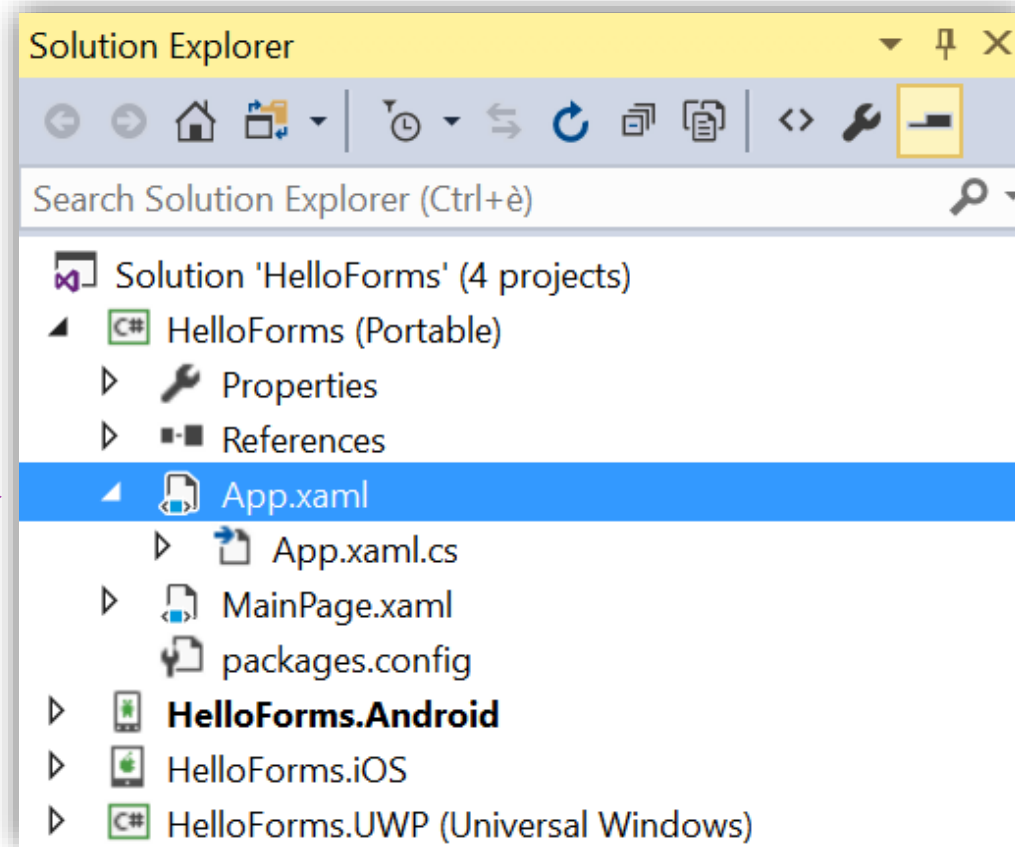




In un progetto Xamarin.Forms la maggior parte del codice platform-independent è confinato nella PCL (o SP)

La classe App è responsabile di:

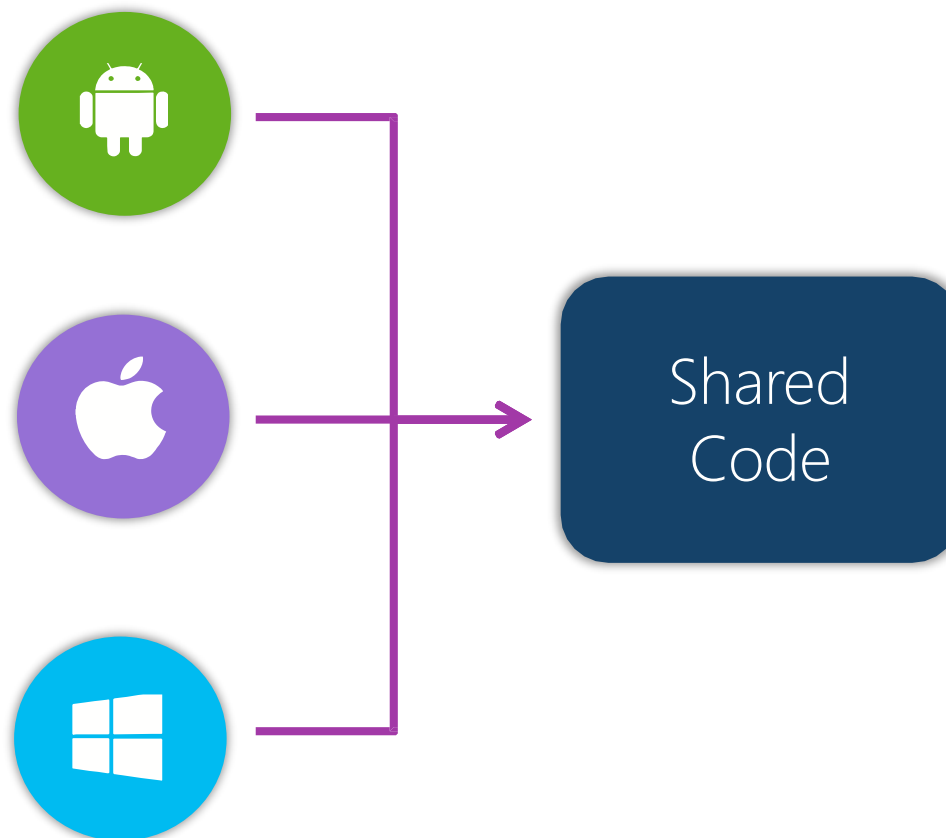
- Avviare la prima pagina dell'applicazione (App.xaml.cs)
- Dichiarare risorse a livello di applicazione (App.xaml)





Progetti platform-specific dipendono dal codice condiviso nella PCL o SP e non viceversa

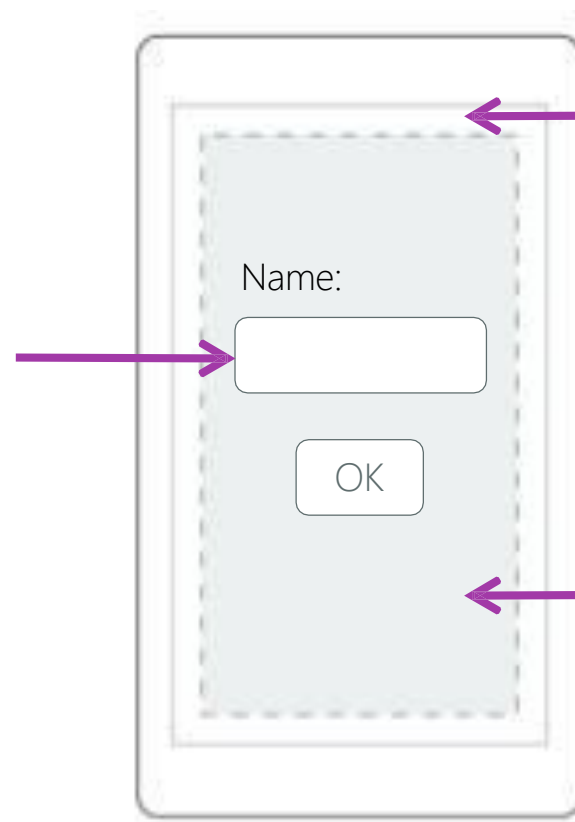
Xamarin.Forms permette di progettare la UI e caratterizzarne il comportamento nella PCL o SP ed eventualmente richiamarne i componenti dai progetti specifici





La UI dell'applicazione è definita in termini di elementi visuali chiamati Page, View e Layout

View: Controllo grafico non composito con cui l'utente può interagire



Page: Singola schermata dell'applicazione

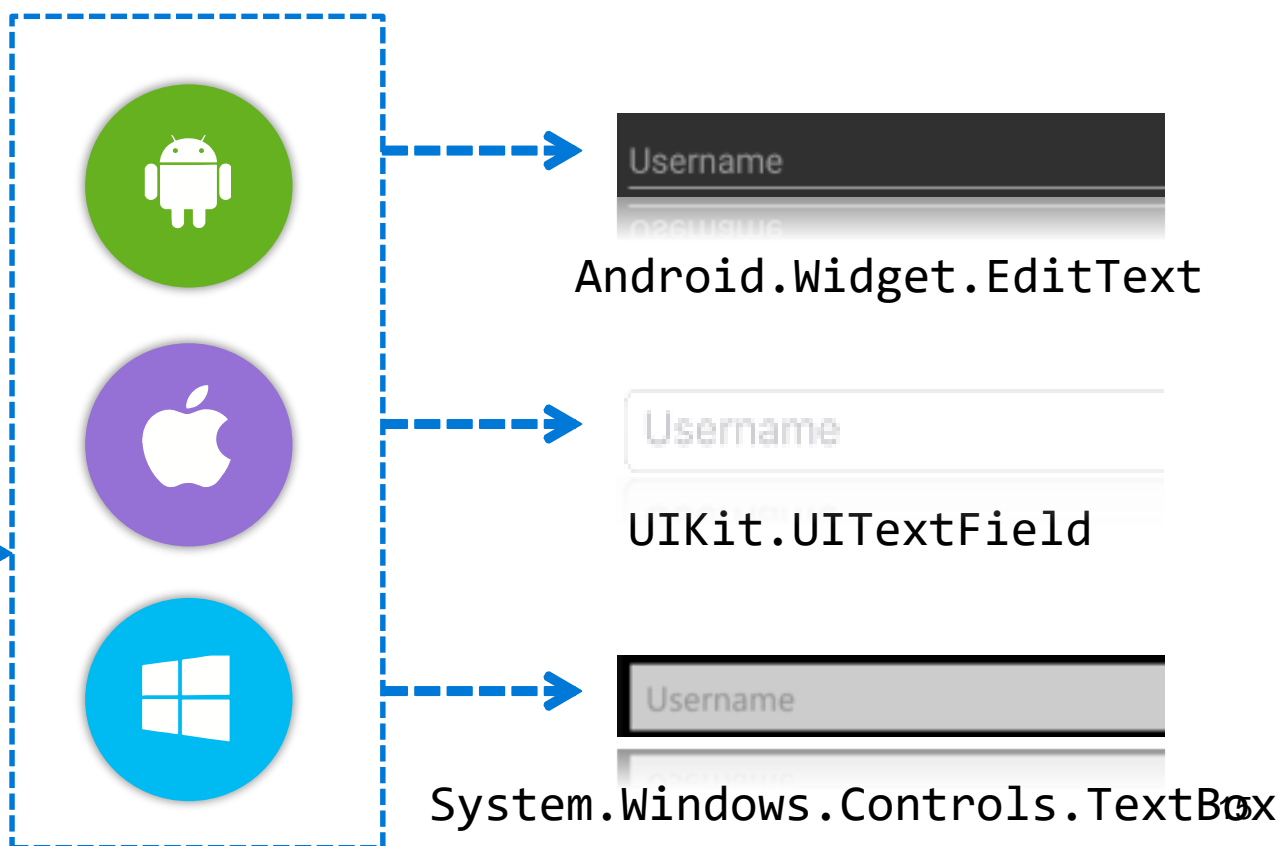
Layout: Contenitori per una o più View

Xamarin.Forms definisce un **Renderer** per ogni Elemento che possiede una rappresentazione nativa nella piattaforma specifica

e.g. `Xamarin.Forms.Entry`

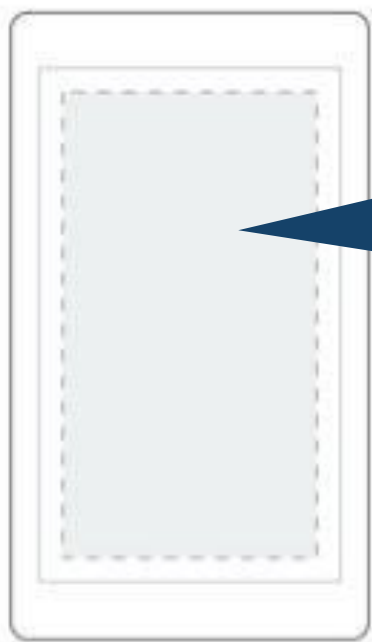
```
Entry entry = new Entry {  
    PlaceholderText = "Username"  
};
```

L'Engine di Rendering traduce la View astratta in un controllo platform-specific

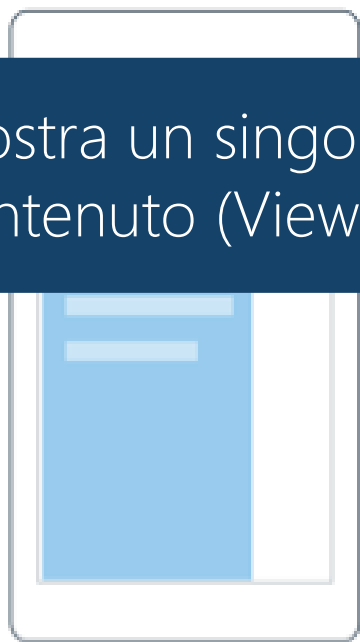




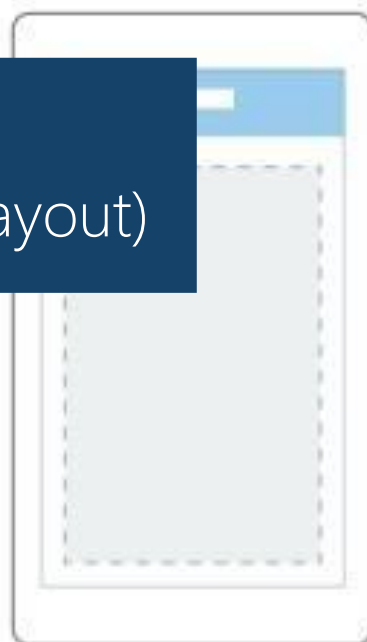
Page espone le funzionalità basilari per rappresentare il contenuto di una singola schermata. Le sue concretizzazioni:



ContentPage



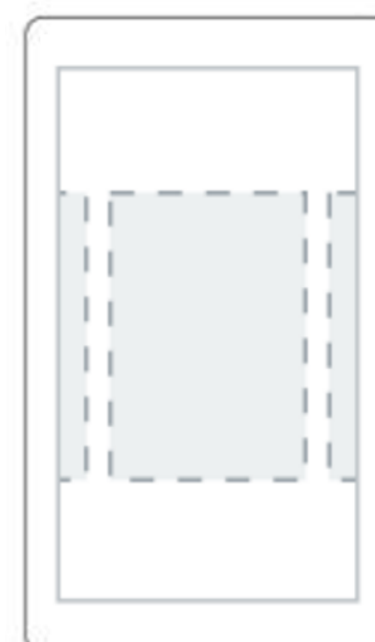
MasterDetailPage



NavigationPage



TabbedPage



CarouselPage

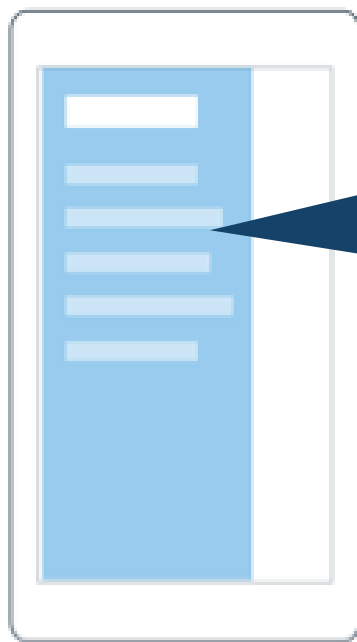
Mostra un singolo contenuto (View o Layout)



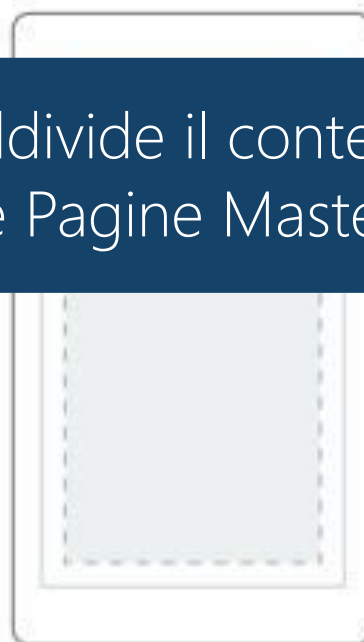
Page espone le funzionalità basilari per rappresentare il contenuto di una singola schermata. Le sue concretizzazioni:



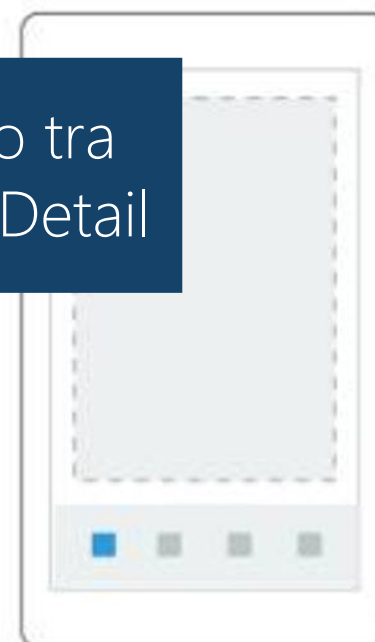
ContentPage



MasterDetailPage



NavigationPage



TabbedPage



CarouselPage

Suddivide il contenuto tra due Pagine Master e Detail

MasterDetailPage viene renderizzata nei corrispondenti elementi nativi



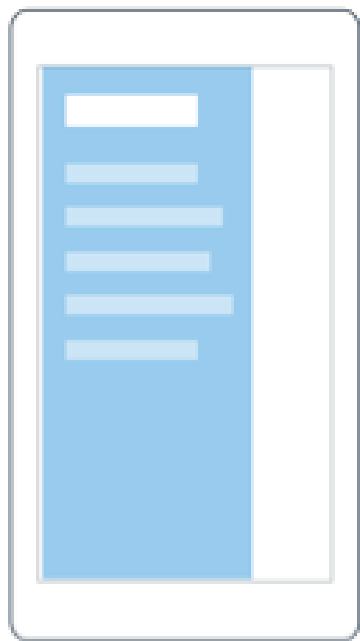
<MasterDetailPage />



Page espone le funzionalità basilari per rappresentare il contenuto di una singola schermata. Le sue concretizzazioni:



ContentPage

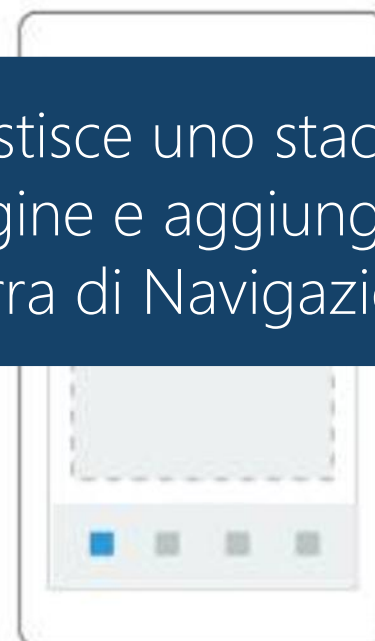


MasterDetailPage



NavigationPage

Gestisce uno stack di
Pagine e aggiunge una
barra di Navigazione



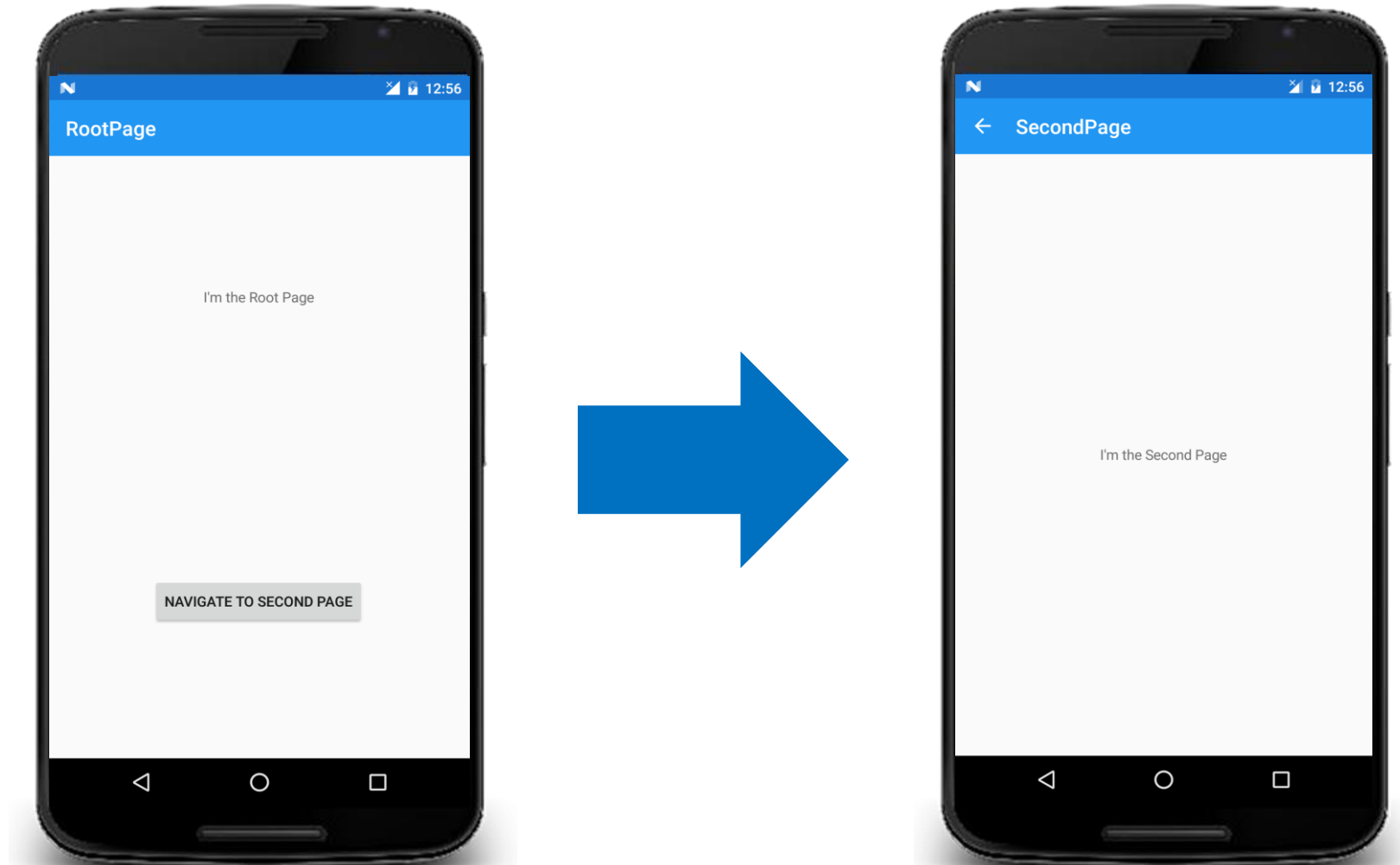
TabbedPage



CarouselPage

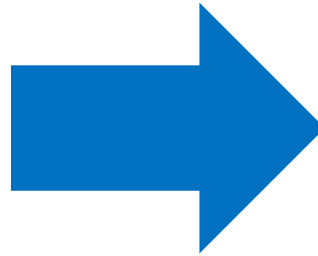
NavigationPage - Android

Introduzione



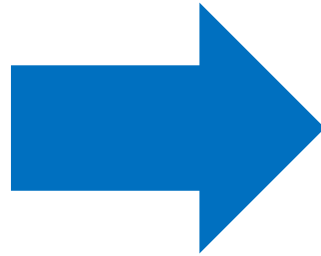
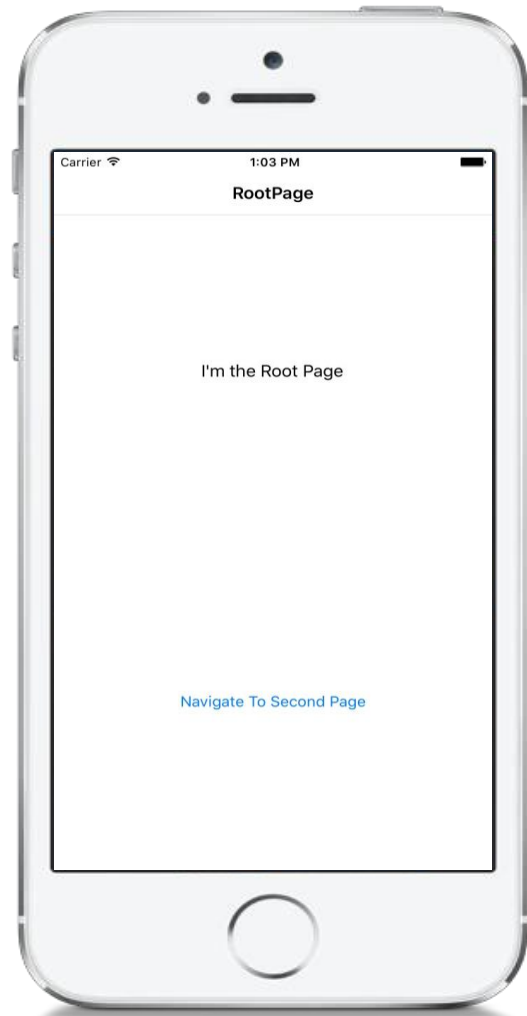
NavigationPage - UWP

Introduzione



NavigationPage - iOS

Introduzione

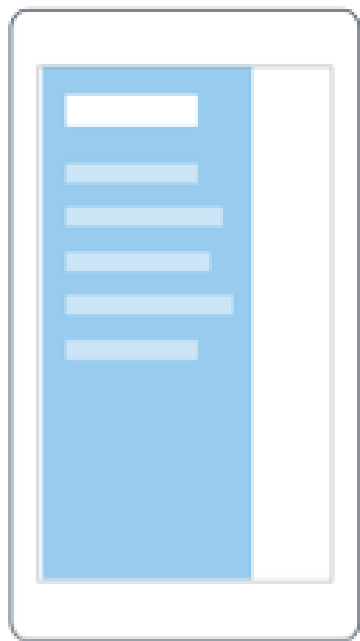




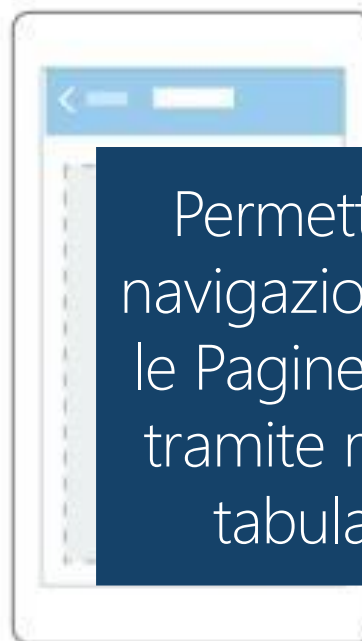
Page espone le funzionalità basilari per rappresentare il contenuto di una singola schermata. Le sue concretizzazioni:



ContentPage



MasterDetailPage



NavigationPage



TabbedPage



CarouselPage

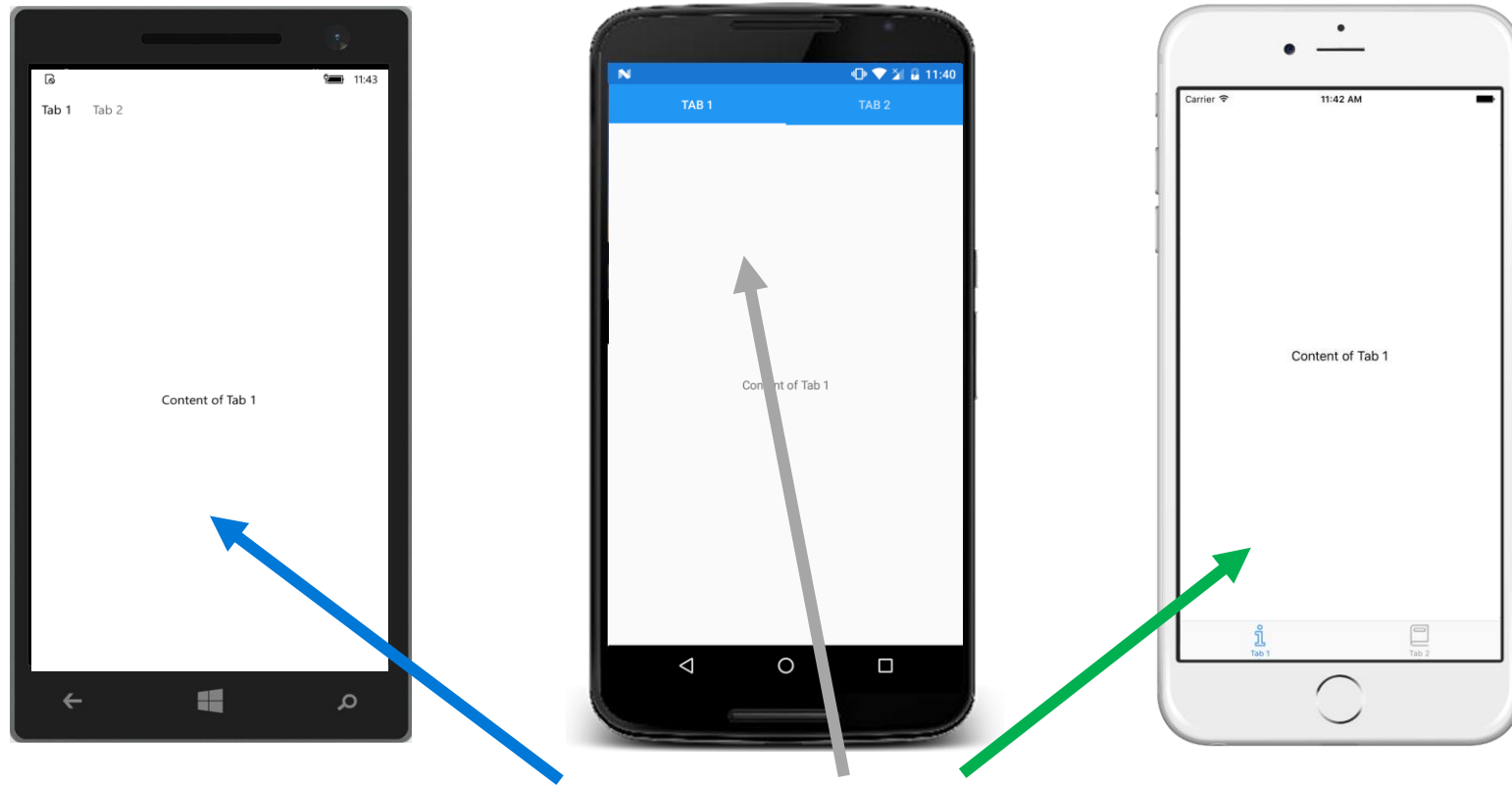
Permette la navigazione tra le Pagine figlie tramite menù tabulare

TabbedPage - Rendering

Introduzione



TabbedPage viene renderizzata nei corrispondenti elementi nativi



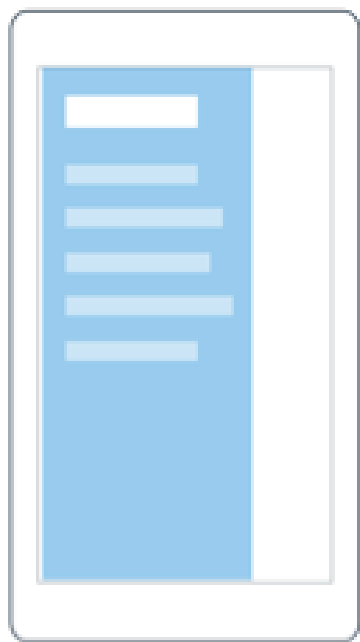
`<TabbedPage />`



Page espone le funzionalità basilari per rappresentare il contenuto di una singola schermata. Le sue concretizzazioni:



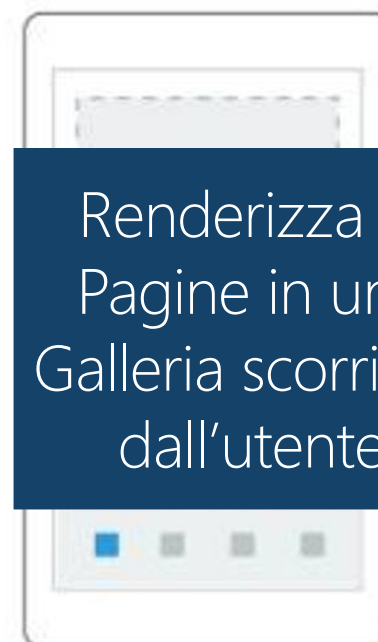
ContentPage



MasterDetailPage



NavigationPage



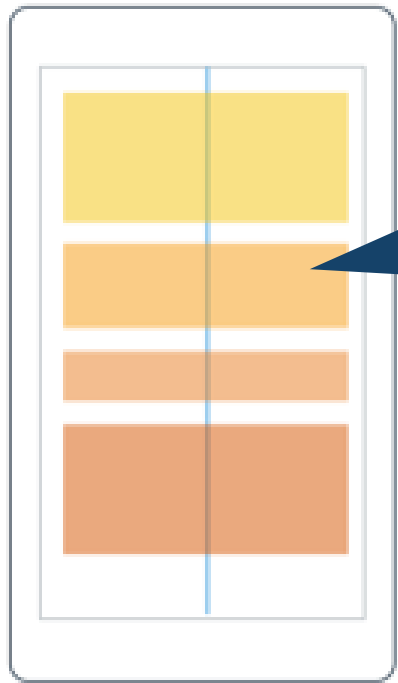
TabbedPage



CarouselPage

Renderizza le
Pagine in una
Galleria scorribile
dall'utente

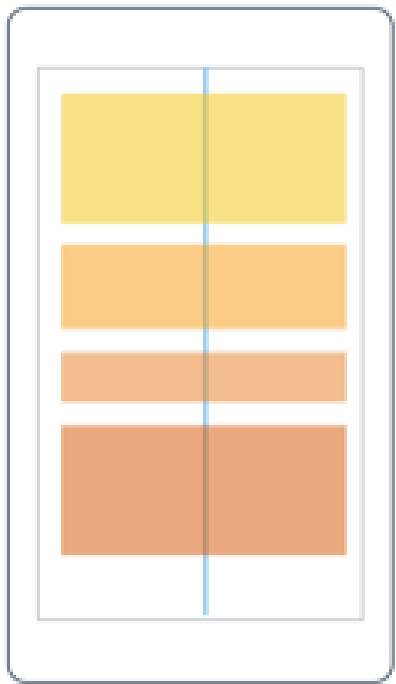
I **Layout** sono contenitori che organizzano posizione e dimensione delle subview sulla base di regole specifiche



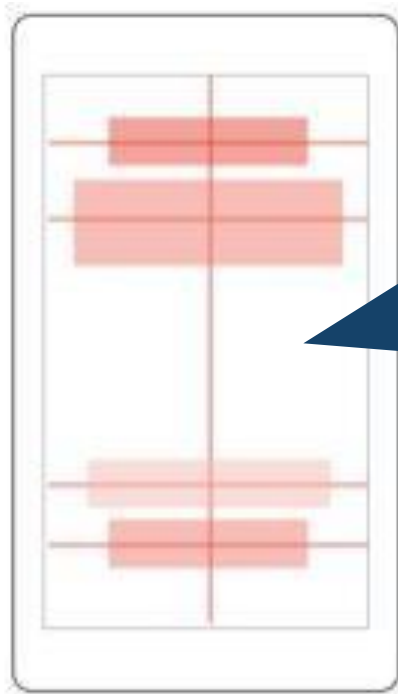
Dispone in cascata le View orizzontalmente o verticalmente

StackLayout

I **Layout** sono contenitori che organizzano posizione e dimensione delle subview sulla base di regole specifiche



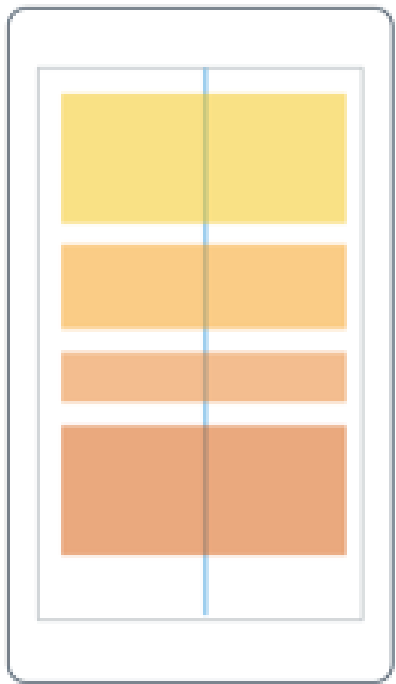
StackLayout



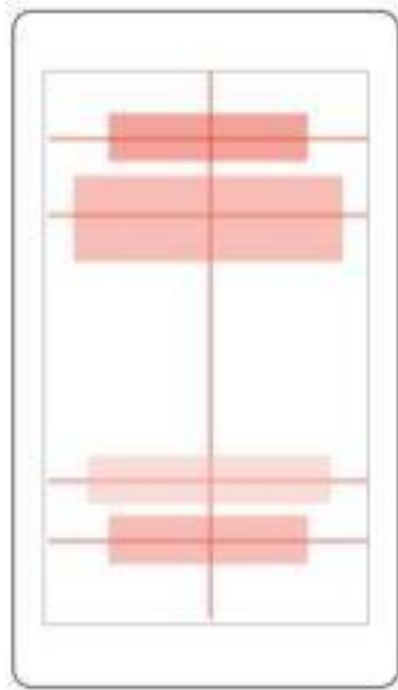
Absolute
Layout

Posiziona subview in
maniera assoluta a
partire da Punti o
Rettangoli

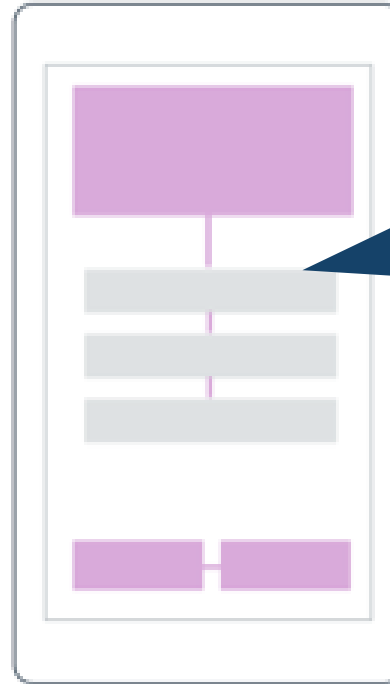
I **Layout** sono contenitori che organizzano posizione e dimensione delle subview sulla base di regole specifiche



StackLayout



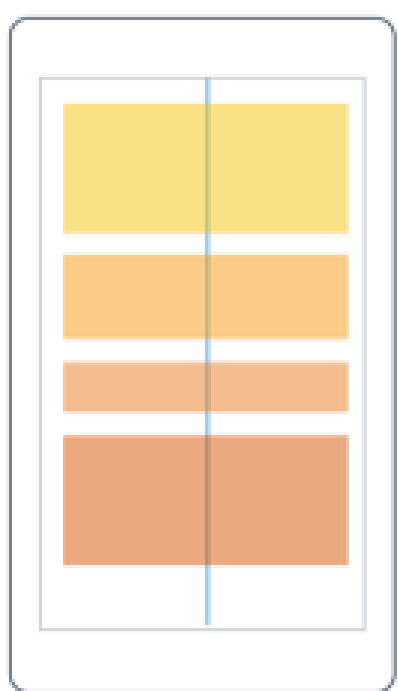
Absolute
Layout



Relative
Layout

Utilizza un sistema di constraint per posizionare View figlie

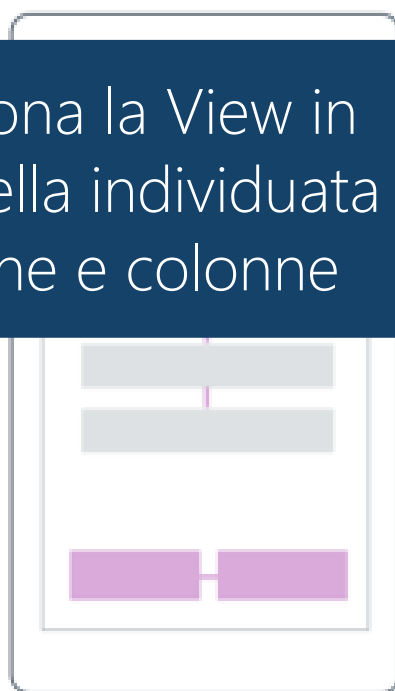
I **Layout** sono contenitori che organizzano posizione e dimensione delle subview sulla base di regole specifiche



StackLayout



Absolute
Layout



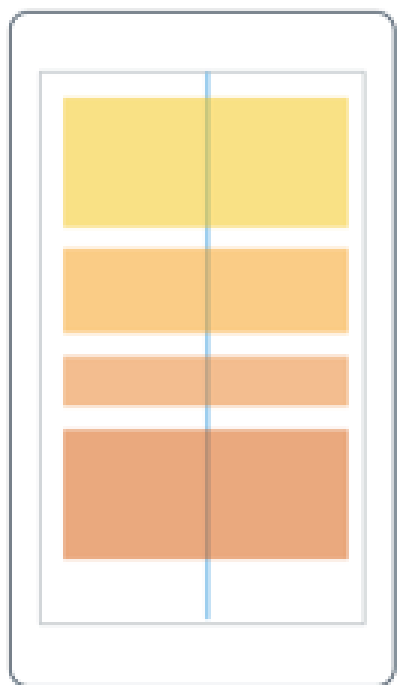
Relative
Layout



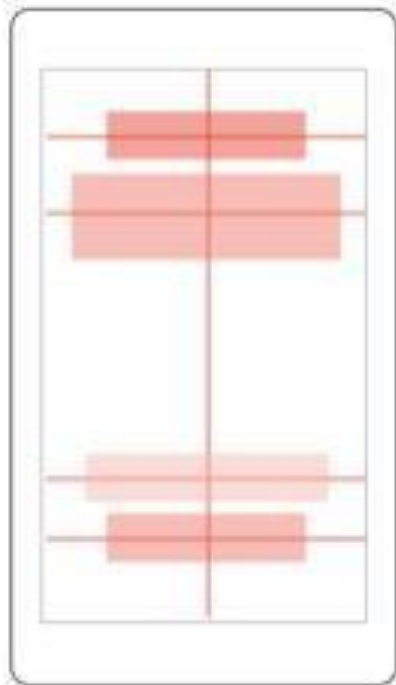
Grid

Posiziona la View in
una cella individuata
da righe e colonne

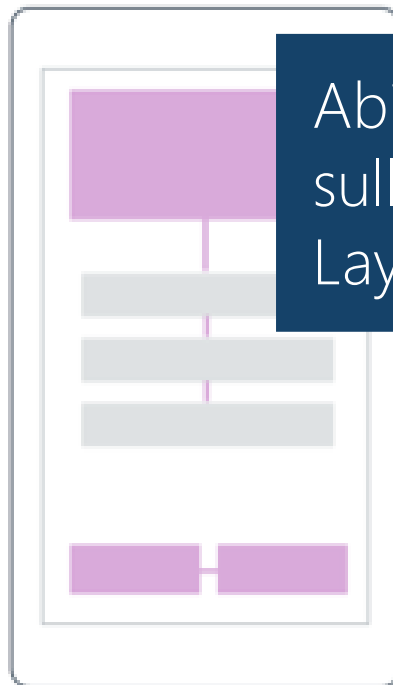
I **Layout** sono contenitori che organizzano posizione e dimensione delle subview sulla base di regole specifiche



StackLayout



Absolute
Layout



Relative
Layout



Grid



ScrollView

Abilita lo scrolling
sulla singola View o
Layout figlio



ActivityIndicator

BoxView

Button

DatePicker

Editor

Entry

Image

Label

ListView

Map

OpenGLView

Picker

ProgressBar

SearchBar

Slider

Stepper

TableView

TimePicker

WebView

EntryCell

ImageCell

SwitchCell

TextCell

ViewCell



Design della UI legato a un linguaggio di programmazione

UI e Comportamento sono definiti in unico file (highly-coupled) → non favorisce la suddivisione dei compiti tra sviluppatore e designer

```
public MainPage()
{
    var rootLayout = new StackLayout();

    _billEntry = new Entry()
    {
        Placeholder = "Enter Bill",
        PlaceholderColor = Color.Accent,
        Keyboard = Keyboard.Numeric,
        HorizontalTextAlignment = TextAlignment.Center
    };
    var calculateButton = new Button()
    {
        Text = "CALCULATE",
        HorizontalOptions = LayoutOptions.Fill
    };
    calculateButton.Clicked += CalculateButtonOnClicked;

    rootLayout.Children.Add(_billEntry);
    rootLayout.Children.Add(calculateButton);

    var tipLayout = new StackLayout
    {
        Orientation = StackOrientation.Horizontal
    };
    tipLayout.Children.Add(new Label
    {
        Text = "Tip",
        FontSize = Device.GetNamedSize(NamedSize.Medium, typeof(Label))
    });
}
```



Sviluppare la UI con un linguaggio di markup (HTML, XML e derivati) porta a numerosi benefici:

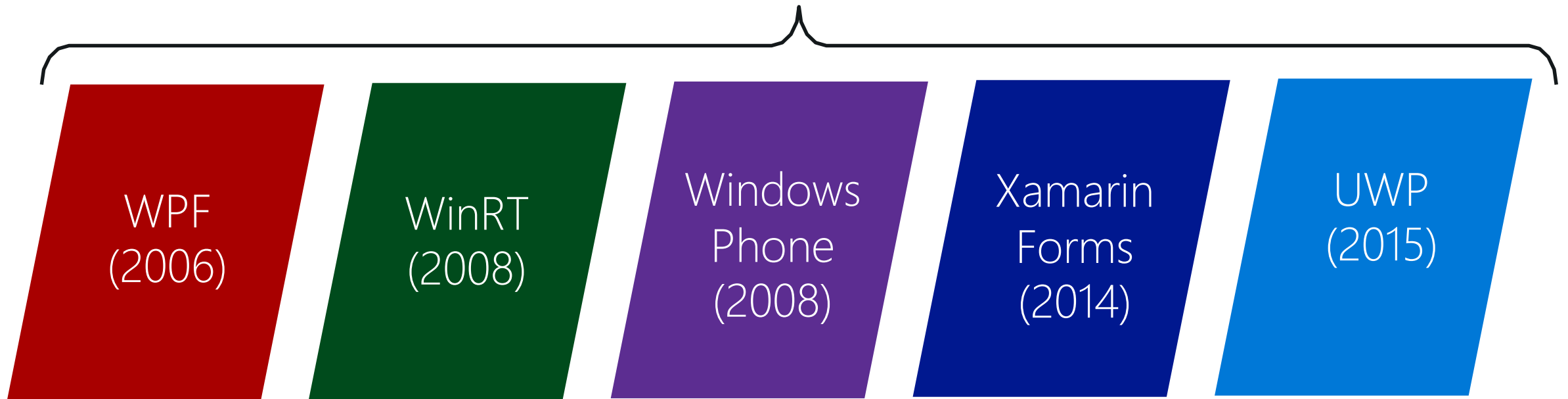
- Portabilità → Pura struttura slegata dal comportamento
- Estendibilità → Si possono creare tag personalizzati
- Leggibilità → Struttura ad albero facilmente leggibile

UI e Comportamento loosely-coupled - Designer vs. Sviluppatore



Creato da Microsoft per descrivere la struttura della UI di un'applicazione

Extensible Application Markup Language



Microsoft XAML vs. Xamarin.Forms

Introduzione



Xamarin.Forms è conforme alla specifica [MS-XAML-2009](#)

Le uniche differenze sono nei nomi dei tag - controlli e layout utilizzabili

Basato su marcatore XML: case sensitive, open/close tag, schema definito

```
<Page x:Class="HelloForms.UWP.MainPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:local="using:HelloForms.UWP"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      mc:Ignorable="d">
  <StackPanel>
    <TextBox PlaceholderText="Email"/>
    <PasswordBox PlaceholderText="Password"/>
    <Button Content="Sign Up!"
            Foreground="Fuchsia"/>
  </StackPanel>
</Page>
```

Microsoft XAML (UWP)

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:local="clr-namespace:HelloForms"
              x:Class="HelloForms.MainPage">
  <StackLayout Spacing="20" Padding="20">
    <Entry Placeholder="Email"/>
    <Entry Placeholder="Password"
            IsPassword="True"/>
    <Button Text="Sign Up!"
            TextColor="Fuchsia"/>
  </StackLayout>
</ContentPage>
```

Xamarin.Forms

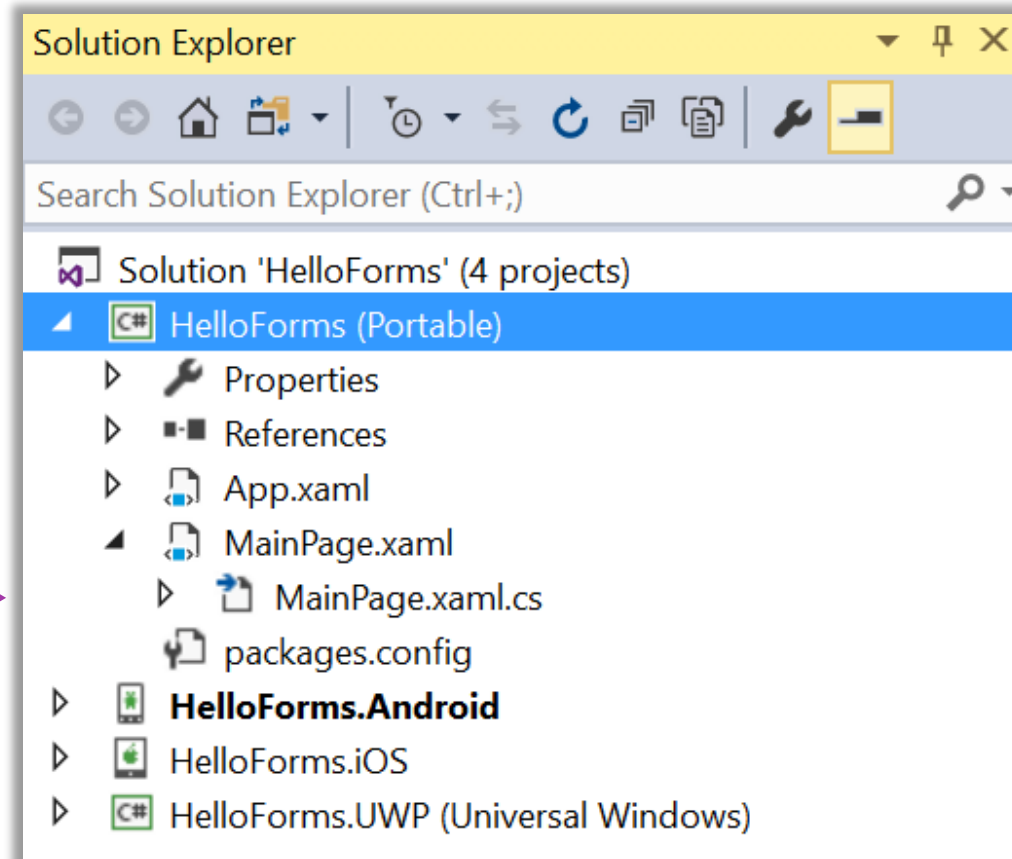
Struttura di una Pagina XAML

Introduzione



Due file contribuiscono alla definizione della classe nella PCL

C# file -
code-behind



XAML file (UI)



Il markup XAML è utilizzato per costruire l'albero degli elementi visuali

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```

Descrizione di una Pagina - Elementi

Introduzione



Il markup XAML è utilizzato per costruire l'albero degli elementi visuali

Gli elementi
XML sono
mappati in
oggetti .NET



```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```



Il markup XAML è utilizzato per costruire l'albero degli elementi visuali

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```

Gli attributi XML sono mappati in proprietà o eventi .NET

Descrizione di una Pagina - Children

Introduzione



Il markup XAML è utilizzato per costruire l'albero degli elementi visuali

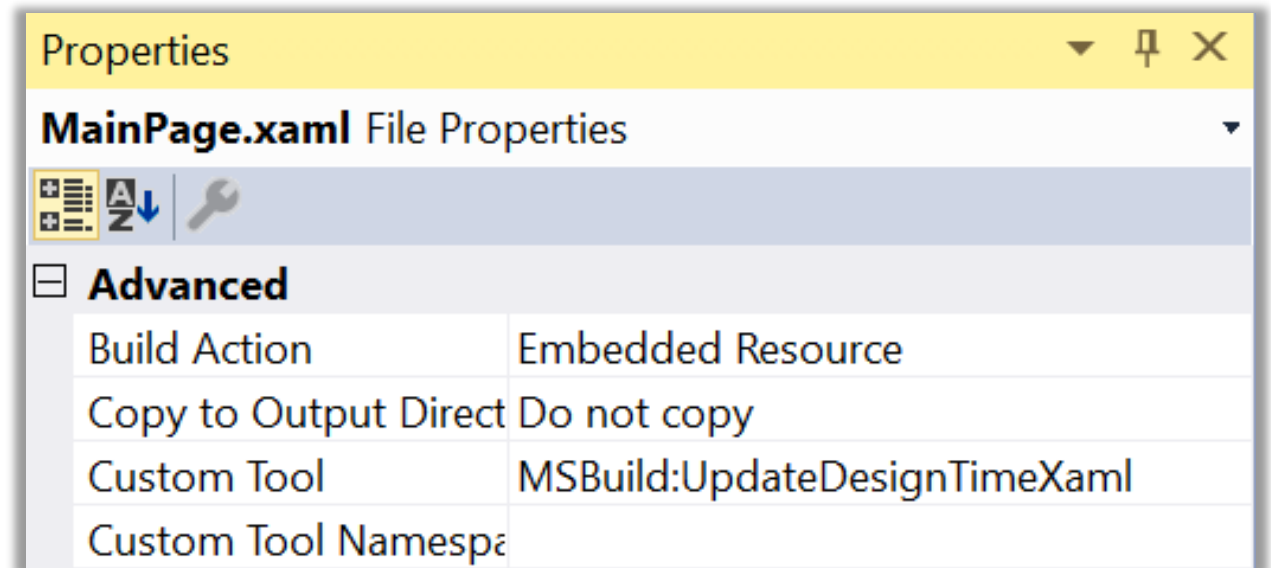
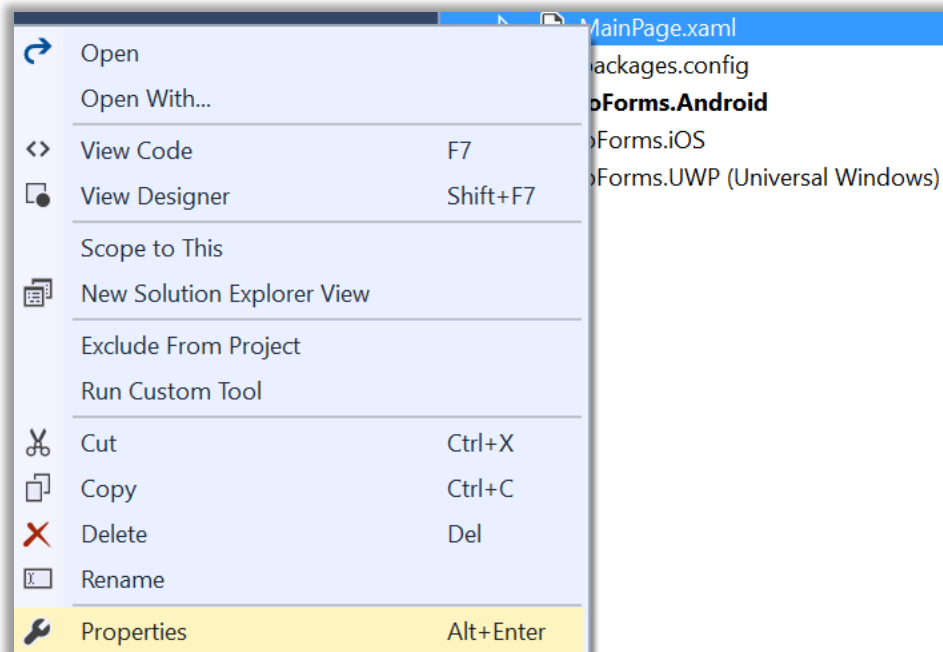
Nodi XML
innestati
vengono
mappati come
Children del
Layout



```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```



I file XAML sono etichettati come "Embedded Resource" e compilati specificando l'opzione **MSBuild:UpdateDesignTimeXaml**





XAML e code-behind sono compilati assieme sfruttando il tag **x:Class** e la definizione parziale della classe

```
<?xml version="1.0" encoding="UTF-8" ?>  
<ContentPage x:Class="HelloForms.MainPage" ...>
```

```
namespace HelloForms  
{  
    public partial class MainPage : ContentPage  
    {  
        ...  
    }  
}
```



Il costruttore pubblico nel code-behind effettua la chiamata al metodo **InitializeComponent**, responsabile di caricare lo XAML e istanziare i suoi oggetti

```
public partial class MainPage : ContentPage
{
    public MainPage ()
    {
        InitializeComponent ();
    }
}
```

Generato dal
Compilatore XAML

```
public partial class MainPage : ContentPage
{
    [GeneratedCodeAttribute(
        "Xamarin.Forms.Build.Tasks.XamlG", "0.0.0.0")]
    private void InitializeComponent()
    {
        this.LoadFromXaml(typeof(MainPage));
    }
}
```

Identificare Elementi in XAML

Introduzione



Utilizzare l'attributo **x:Name** per assegnare un nome all'elemento XAML così da poterlo referenziare da code-behind



Compile-time viene aggiunto un campo privato alla classe parziale generata in fase di compilazione (.g.cs)

MainPage.xaml

```
<Entry x:Name="EmailEntry"
        Placeholder="Email" />
```

```
public partial class MainPage : ContentPage
{
    private Entry EmailEntry;

    private void InitializeComponent() {
        this.LoadFromXaml(typeof(MainPage));
        EmailEntry = this.FindByName<Entry>(
            "EmailEntry");
    }
}
```

MainPage.xaml.g.cs
44



È possibile utilizzare elementi XAML con nome come fossero oggetti normalmente definiti in C# solo dopo la chiamata al metodo

InitializeComponent

Si possono associare event handler, modificare proprietà o anche integrare elementi definiti da codice

```
public partial class MainPage : ContentPage
{
    public MainPage () {
        InitializeComponent ();
        EmailEntry.TextChanged += OnTextChanged;
    }

    void OnTextChanged(object sender, TextChangedEventArgs e) {
        ...
    }
}
```



Si possono gestire eventi da XAML - il nome dell'handler deve corrispondere ad un metodo delegato C# con firma opportuna nel code-behind

```
<Entry Placeholder="Number" TextChanged="OnTextChanged" />
```

```
public partial class MainPage : ContentPage
{
    ...
    void OnTextChanged(object sender, TextChangedEventArgs e) {
        ...
    }
}
```



Così come in C#, è possibile modificare la UI in accordo con la Piattaforma sottostante utilizzando **Device.OnPlatform**

`x:TypeArguments` usato per tipi generici

```
<StackLayout Spacing="10">  
  <StackLayout.Padding>  
    <OnPlatform x:TypeArguments="Thickness"  
      iOS="0,20,0,0" Android="0" Windows="0" />  
  </StackLayout.Padding>  
  ...  
</StackLayout>
```

Valori platform-specific per l'attributo Padding (Property Syntax)



<Flash Quiz/>

Domanda 1

Flash Quiz #1



- 1 Il codice di un'app Xamarin.Forms può essere sempre condiviso tra più piattaforme
 - a) Vero
 - b) Falso

Domanda 1

Flash Quiz #1



- 1 Il codice di un'app Xamarin.Forms può essere sempre condiviso tra più piattaforme
 - a) Vero
 - b) Falso



- 2 In quale di questi casi l'approccio Xamarin.Forms è da preferire all'approccio classico Xamarin.Native
- a) Requisito su elevate performance e molte animazioni, transizioni...
 - b) Il Cliente ha esplicitamente chiesto un'app Xamarin.Forms
 - c) L'app è un semplice gestionale per visualizzare dati provenienti da DB
 - d) Si deve consegnare al cliente un prototipo di app Xamarin e il tempo scarseggia



- 2 In quale di questi casi l'approccio Xamarin.Forms è da preferire all'approccio classico Xamarin.Native
- a) Requisito su elevate performance e molte animazioni, transizioni...
 - b) Il Cliente ha esplicitamente chiesto un'app Xamarin.Forms
 - c) L'app è un semplice gestionale per visualizzare dati provenienti da DB
 - d) Si deve consegnare al cliente un prototipo di app Xamarin e il tempo scarseggia



- 3 Quale tra queste non è una Pagina esposta dal framework Xamarin.Forms?
- a) TabbedPage
 - b) CarouselPage
 - c) DataGridPage
 - d) NavigationPage



- 3 Quale tra queste non è una Pagina esposta dal framework Xamarin.Forms?
- a) TabbedPage
 - b) CarouselPage
 - c) DataGridPage
 - d) NavigationPage



- 4 Quale tra questi è un Layout Xamarin.Forms?
- a) StackPanel
 - b) RelativeLayout
 - c) Grid
 - d) Canvas

Domanda 4

Flash Quiz #1



- 4 Quale tra questi è un Layout Xamarin.Forms?
- a) StackPanel
 - b) RelativeLayout
 - c) Grid
 - d) Canvas

Domanda 5

Flash Quiz #1



- 5 Quale tra queste è una View Xamarin.Forms?
- a) RadialGauge
 - b) Entry
 - c) EditText
 - d) UITextField

Domanda 5

Flash Quiz #1



- 5 Quale tra queste è una View Xamarin.Forms?
- a) RadialGauge
 - b) Entry
 - c) EditText
 - d) UITextField



Xamarin

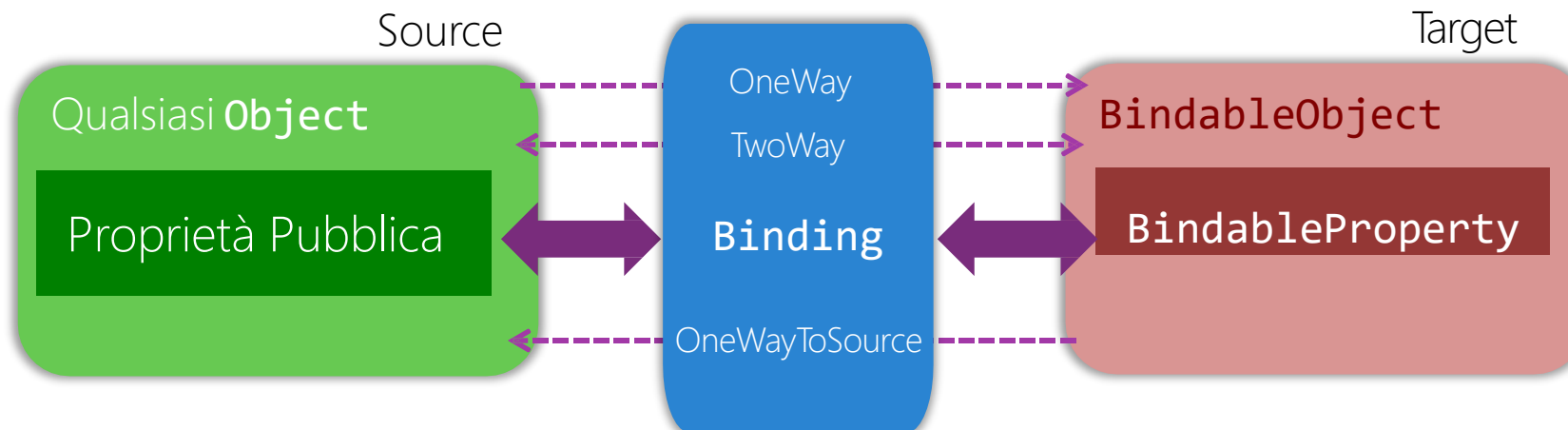
Xamarin.Forms

Data Binding e MVVM

Creare associazioni loosely-coupled tra view e dati

Il **Data Binding** è una funzionalità già integrata in Xamarin.Forms:

- Permette di creare associazioni loosely-coupled tra una proprietà Sorgente e una proprietà Target
- Svincola lo sviluppatore dal dover implementare un Controller per collegare Vista e Modello



Creare Binding - C#

Data Binding e MVVM



```
Profile profile = new Profile() { Name = "Francesco", ... };
```

```
Entry nameEntry = new Entry();
```

```
Binding nameBinding = new Binding();
```

```
nameBinding.Source = profile;
```

```
nameBinding.Path = "Name";
```

```
nameEntry.SetBinding(Entry.TextProperty, nameBinding);
```





È possibile creare binding via XAML utilizzando l'apposita estensione di markup **BindingExtension**

```
<StackLayout Padding="0,50,0,0">
  <StackLayout.Resources>
    <ResourceDictionary>
      <Profile x:Key="myProfile" Name="Francesco" .../>
    </ResourceDictionary>
  </StackLayout.Resources>
  <Entry Text="{Binding Name,
    Source={StaticResource myProfile}}" />
  ...
</StackLayout>
```

Path come primo argomento

Assegnamento a proprietà **Target**

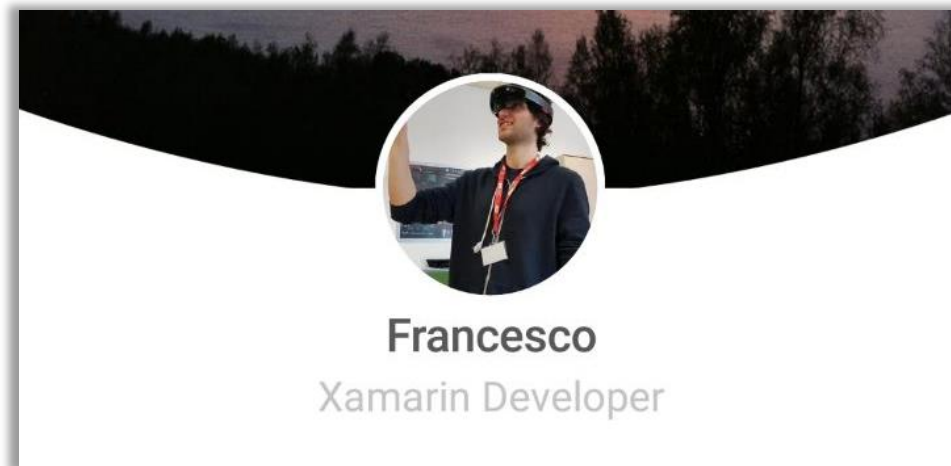
Source fornito come Risorsa Statica



Una pagina mostra diverse proprietà appartenenti a classi modello

```
public class Profile
{
    public string Name { get; set; }
    public string BackgroundImage { get; set; }
    public string ProfileImage { get; set; }
    public string Title { get; set; }
}
```

È possibile specificare la sorgente del Binding ogni volta o utilizzare il **BindingContext** come sorgente di default





La proprietà **BindingContext** rappresenta la sorgente per ogni Binding quando la proprietà **Source** non viene specificata

```
<ContentPage ...  
    BindingContext= -----> Qualsiasi Object  
    "{Binding Source={StaticResource myProfile}}">  
    ...  
    <StackLayout Padding="0,50,0,0">  
        <Label Text="{Binding Name}" />  
        <Label Text="{Binding Title}" />  
    </StackLayout>  
</ContentPage>
```

↓
Nessun Source



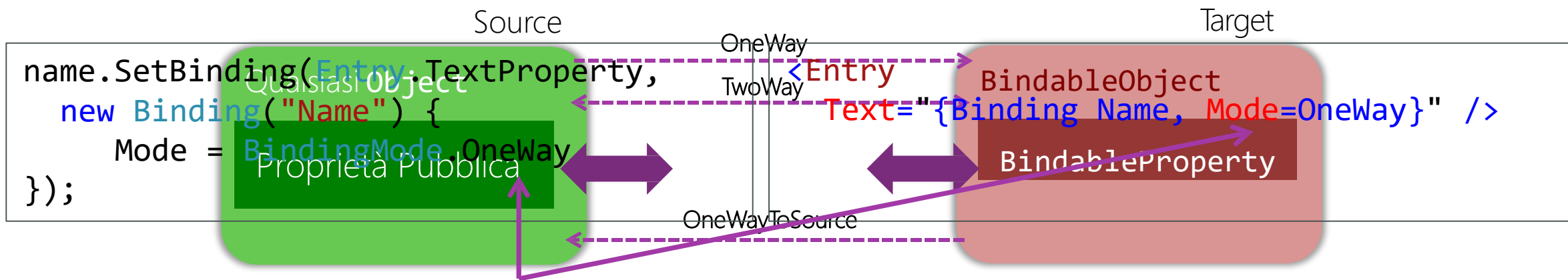
Il **BindingContext** viene automaticamente ereditato dai nodi figlio di un contenitore – se specificato per la radice del VT è ereditato da tutti i figli

Modalità di Binding

Data Binding e MVVM



La proprietà **Mode** di **Binding** permette di controllare la direzione di trasferimento dei dati (e.g. TwoWay per ambo le direzioni)



BindingMode.Default rappresenta la direzione di default del Binding ed il suo valore dipende dalla proprietà Target del controllo

Binding di tipo **OneWay** e **TwoWay** aggiornano la UI ogni volta che cambia il valore della proprietà sorgente

```
profile.Name = "Satya";  
//...
```

```
public class Profile  
{  
    public string Name { get; set; }  
    public string BackgroundImage { get; set; }  
    public string ProfileImage { get; set; }  
    public string Title { get; set; }  
}
```





L'interfaccia .NET **INotifyPropertyChanged** consente di "stipulare" il contratto di notifica delle proprietà di un'entità

```
namespace System.ComponentModel
{
    public interface INotifyPropertyChanged
    {
        event PropertyChangedEventHandler PropertyChanged;
    }
}
```



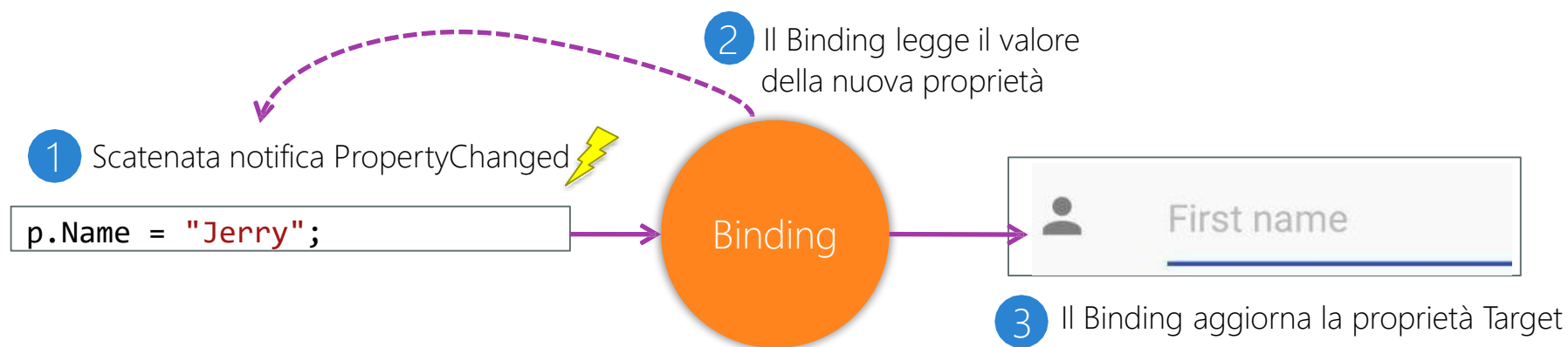
Binding di tipo **TwoWay** sono gestiti automaticamente dal **BindableObject** Target il quale implementa già l'interfaccia **INotifyPropertyChanged**

```
public abstract class BindableObject : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;
    private void SetValue(BindableProperty property/* ... */)
    {
        // ...
        PropertyChanged((object)this,
            new PropertyChangedEventArgs(property.Name));
    }
}

[TypeConverter(typeof(BindablePropertyConverter))]
public sealed class BindableProperty {
    //...
}
```

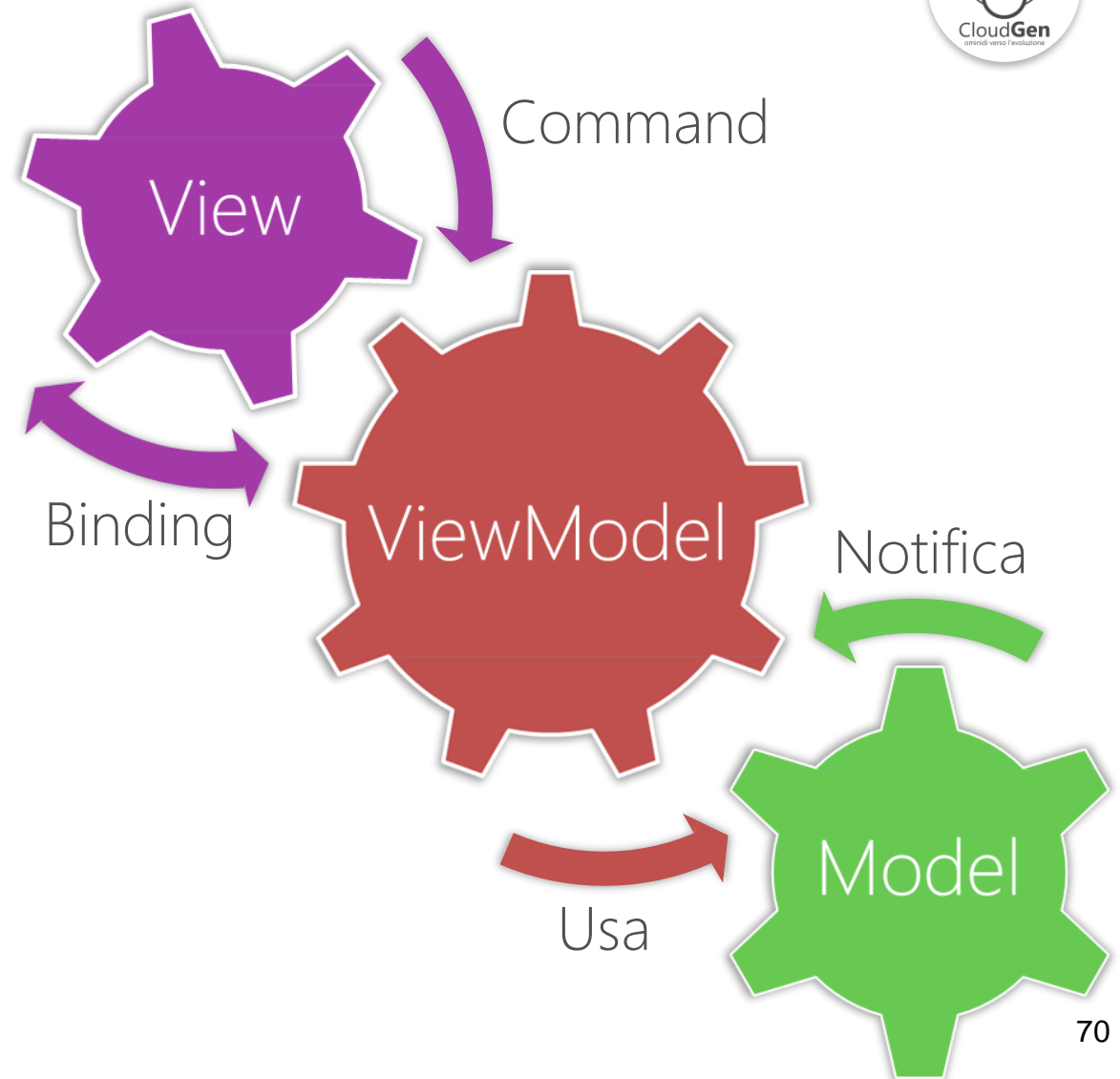


Il **Binding** sottoscrive l'evento **PropertyChanged** e ogniqualvolta viene notificato di una modifica della proprietà sorgente aggiorna conseguentemente la proprietà Target



Model-View-ViewModel (MVVM) è un design pattern architetturale (o di framework) atto a separare la logica dell'applicazione tra UI, dati (più comunemente modello) e comportamento

Il ViewModel prende il posto del controller di MVC gestendo l'interazione tra View e Model grazie all'engine di Data Binding





Il **Model** contiene il modello dei dati e la logica di business dell'applicazione (e.g. Persistenza, Validazione...)

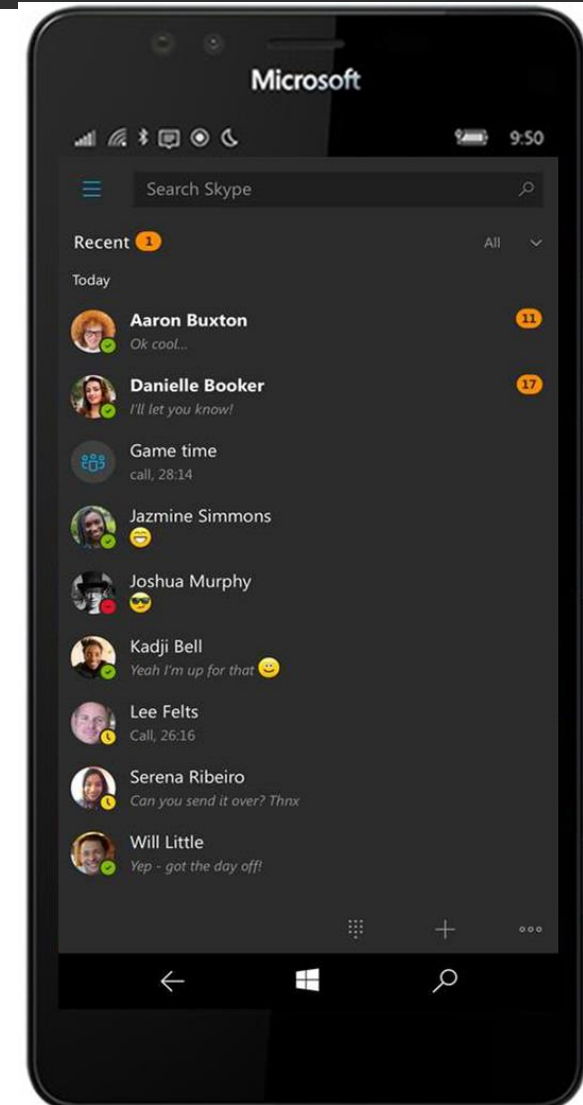
Il Model fa parte dello Shared Code - pertanto non dovrebbe contenere funzionalità specifiche della piattaforma

```
public class Profile {  
    public string Name { get; set; }  
    public string BackgroundImage { get; set; }  
    public string ProfileImage { get; set; }  
    public string Title { get; set; }  
    public string Description { get; set; }  
    public uint Likes { get; set; }  
    public uint Following { get; set; }  
    public uint Followers { get; set; }  
}
```

La **View** presenta le informazioni all'utente utilizzando il look-and-feel nativo della piattaforma

Non dovrebbe contenere codice su cui effettuare unit testing

Proprietà visuali e animazioni devono essere gestite a questo livello





Il **ViewModel** fornisce una rappresentazione view-centrica del **Modello** alla **View**

Esponde le
proprietà
sorgenti per il
Binding

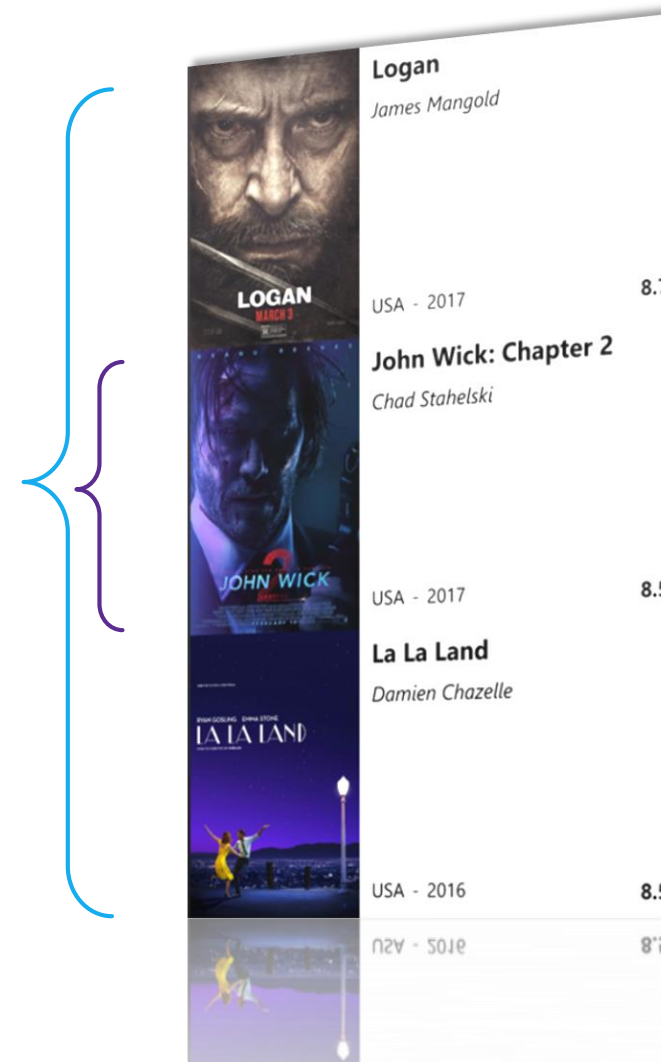
```
public class ProfileViewModel : INotifyPropertyChanged {  
    private readonly Profile _profile;  
    public ProfileViewModel(Profile profile) {  
        _profile = profile;  
    }  
    public string Name {  
        get => _profile.Name;  
        set {  
            if (_profile.Name == value)  
                return;  
            _profile.Name = value;  
            OnPropertyChanged();  
        }  
    }  
}
```

Incapsula il
Modello

Spesso un'applicazione possiede diversi ViewModel – uno per ogni entità per cui si prevede una rappresentazione della UI

Un ViewModel può essere condiviso tra più View (**ItemViewModel**)

In MVVM ad ogni **Page** è associato il corrispondente ViewModel (**PageViewModel**), che agisce da Controller per quella Page





Un **PageViewModel** viene spesso utilizzato come **BindingContext** per l'intera pagina - si può specificare sia tramite C# che XAML

```
public partial class MainPage : ContentPage {  
    private readonly MainPageViewModel _viewModel  
    public MainPage()  
    {  
        BindingContext = _viewModel =  
            new MainPageViewModel();  
        InitializeComponent();  
    }  
    //...  
}
```

```
<ContentPage ...>  
    <ContentPage.BindingContext>  
        <viewModels:MainPageViewModel/>  
    </ContentPage.BindingContext>  
</ContentPage>
```



{Flash Quiz}

Domanda 1

Flash Quiz #2



- 1 La sorgente di un Binding è specificata attraverso _____.
- a) Proprietà DataContext di BindableObject
 - b) Proprietà Source di Binding
 - c) Proprietà BindingContext di BindableObject
 - d) Proprietà Path di Binding

Domanda 1

Flash Quiz #2



- 1 La sorgente di un Binding è specificata attraverso _____.
- a) Proprietà DataContext di BindableObject
 - b) Proprietà Source di Binding
 - c) Proprietà BindingContext di BindableObject
 - d) Proprietà Path di Binding

Domanda 2

Flash Quiz #2



- 2 Utilizzando il pattern MVVM, il ViewModel deve possedere un'implementazione specifica per ogni piattaforma che si intende supportare
- a) Vero
 - b) Falso

Domanda 2

Flash Quiz #2



- 2 Utilizzando il pattern MVVM, il ViewModel deve possedere un'implementazione specifica per ogni piattaforma che si intende supportare
- a) Vero
 - b) Falso



- 3 Quali sono i campi esposti dall'interfaccia INotifyPropertyChanged?
- a) Evento PropertyChanged e metodo RaisePropertyChanged
 - b) Solo Metodo OnPropertyChanged
 - c) Solo Evento PropertyChanged



- 3 Quali sono i campi esposti dall'interfaccia INotifyPropertyChanged?
- a) Evento PropertyChanged e metodo RaisePropertyChanged
 - b) Solo Metodo OnPropertyChanged
 - c) Solo Evento PropertyChanged

Domanda 3

Flash Quiz #2



- 4 Si può effettuare Unit Testing su un ViewModel
- a) Vero
 - b) Falso

Domanda 3

Flash Quiz #2



- 4 Si può effettuare Unit Testing su un ViewModel
- a) Vero
 - b) Falso



- 5 Quale dei seguenti **non** è un Design Pattern architetturale?
- a) MVP
 - b) MVC
 - c) MVPM
 - d) MVVM



- 5 Quale dei seguenti **non** è un Design Pattern architetturale?
- a) MVP
 - b) MVC
 - c) MVPM
 - d) MVVM



Xamarin

Xamarin.Forms

Road to Xamarin.Forms 3.0

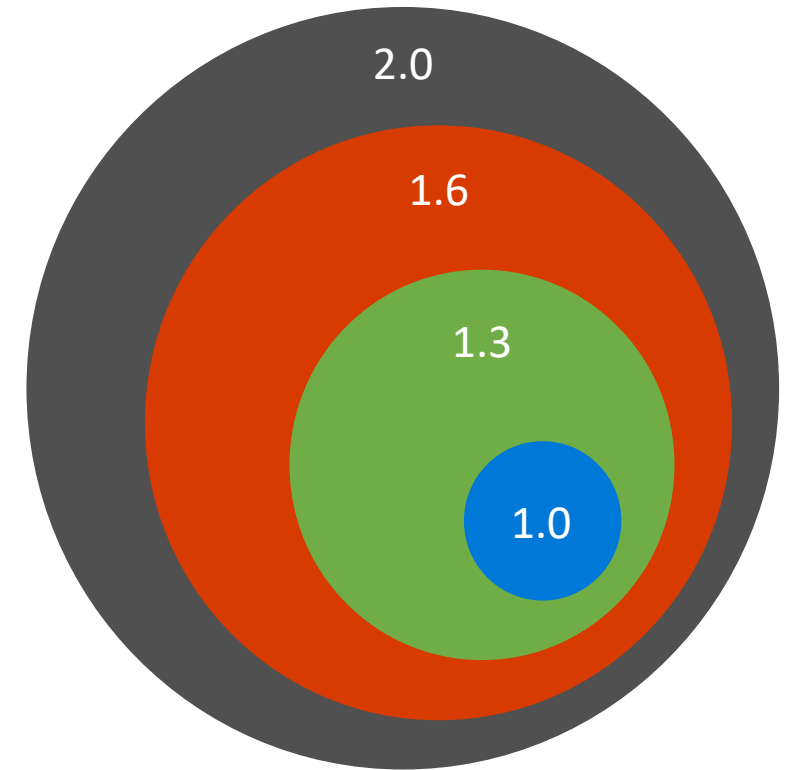
What's new and what's coming next?



Rappresentano un set di API che tutte le piattaforme .NET (.NET Framework, .NET Core, Xamarin) devono implementare

Le API esposte non sono ricavate dall'intersezione tra piattaforme ma sono indipendenti da queste

Hanno versioni lineari – versioni maggiori incorporano le API offerte dalle versioni precedenti



aka.ms/xf-xamlstandard



Proposta di uno Standard per uniformare il nome di controlli e proprietà tra UWP e Xamarin.Forms

XAML attuale

```
<StackLayout Orientation="Horizontal"
              Background="SkyBlue">
  <Label Text="UserName: " />
  <Entry PlaceholderText="Enter your username" />
</StackLayout>
```



XAML Standard 1.0

```
<StackPanel Orientation="Horizontal"
             Background="SkyBlue">
  <TextBlock Text="UserName: " />
  <TextBox PlaceholderText="Enter your username" />
</StackLayout>
```


XAML Standard

Road to Xamarin.Forms 3.0



Proposta di uno Standard per uniformare il nome di controlli e proprietà tra UWP e Xamarin.Forms

aka.ms/xf-xamlstandard

Microsoft / xaml-standard

Unwatch 92 Unstar 575 Fork 27

Code Issues 157 Pull requests 1 Projects 0 Wiki Insights

Filters is:issue is:open Labels Milestones New issue

157 Open 32 Closed Author Labels Projects Milestones Assignee Sort

- Discussion: XAML is more than XML. Code-behind needs consideration too
#193 opened 9 days ago by dotMorten 31
- Add MathMLView Control controls proposal
#192 opened 11 days ago by insinfo 6
- Add CommandParameter to Button
#191 opened 11 days ago by Daniel-Svensson 2
- Add Padding property to Label, TextBox, Button, etc. proposal
#190 opened 12 days ago by dan-meier 6

Forms Embedding - Android

Road to Xamarin.Forms 3.0



Possiamo utilizzare le API attuali per caricare pagine Xamarin.Forms in progetti Xamarin.Native – per View e Layout bisogna però aspettare...

Dopodiché, si avvia normalmente l'Activity con le API di Xamarin.Android passando l'indicazione della pagina Forms nel Bundle Extra

```
public class FormsActivity : FormsAppCompatActivity {  
    protected override void OnCreate(Bundle bundle)  
    {  
        base.OnCreate(bundle);  
        var pageName = Intent.Extras.GetString("PageType");  
        var fullName = typeof(FormsApp).Namespace + ".Pages." + pageName;  
        var pageType = typeof(FormsApp).Assembly.GetType(fullName);  
        if (!IsFormsInitialized) {  
            global::Xamarin.Forms.Init(this, bundle);  
            Forms.Init(this, bundle);  
            Forms.InitExtra("PageType", "SettingsPage");  
        }  
        StartActivity(intent);  
        LoadApplication(new FormsApp(pageType));  
    }  
}
```

```
public FormsApp(Type pageType) {  
    MainPage =  
        (Page)Activator.  
            CreateInstance(pageType);  
}
```

Forms Embedding - UWP

Road to Xamarin.Forms 3.0



Possiamo utilizzare le API attuali per caricare pagine Xamarin.Forms in progetti Xamarin.Native – per View e Layout bisogna però aspettare...

Dopo di che, si utilizzano normalmente le API di Navigazione di UWP passando come parametro di navigazione il tipo della Pagina Forms di destinazione

```
public class FormsWrapperPage : WindowsPage
```

```
{  
    private readonly FormsApp _formsApp;  
    public FormsWrapperPage()
```

```
{  
        this.InitializeComponent();  
        LoadApplication(_formsApp = new FormsApp());  
    }
```

```
protected override void OnNavigatedTo(NavigationEventArgs e)
```

```
{  
        base.OnNavigatedTo(e);  
        _formsApp.SetMainPage(e.Parameter as Type);  
    }
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
public SetMainPage(Type pageType) {
```

```
    MainPage =
```

```
    PageActivator.
```

```
    CreateInstance(pageType);
```

```
}
```



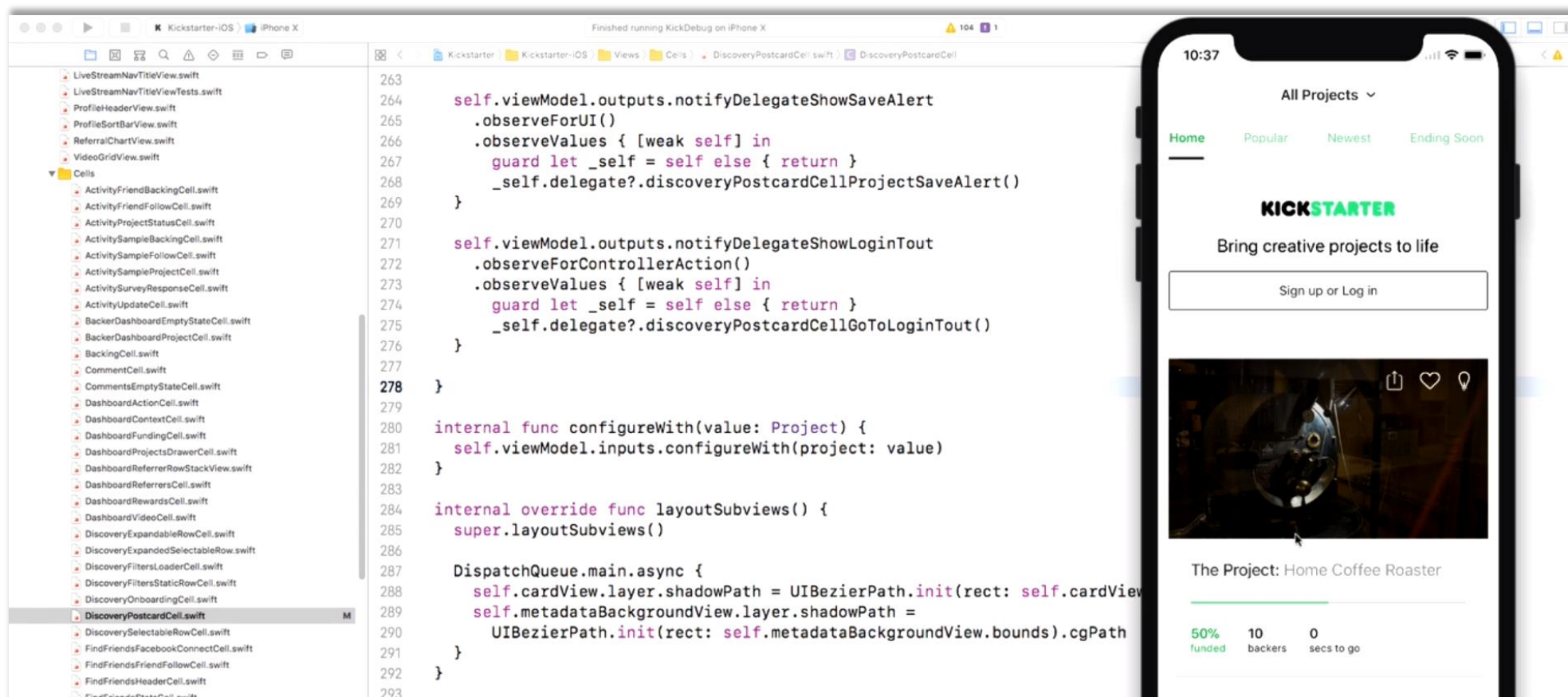
Possiamo utilizzare le API attuali per caricare pagine Xamarin.Forms in progetti Xamarin.Native – per View e Layout bisogna però aspettare...

```
public static Page GetPage<T>() where T : Page
{
    return Activator.CreateInstance<T>();
}
```

Nel caso di iOS la classe `Xamarin.Forms.PageExtensions` espone il metodo di estensione `CreateViewController` → Dopodichè, si può presentare il `ViewController` utilizzando le API di `Xamarin.iOS`

```
var settingsViewController = FormsApp.GetPage<SettingsPage>().CreateViewController();
await this.PresentViewControllerAsync(settingsViewController, true);
```

Possibilità di consumare componenti .NET a partire da applicazioni native Android (Java), iOS (Obj-C) e Desktop (C)





In Android, abilitare i Fast Renderer permette di diminuire la densità del VT nativo per le View **Button**, **Image**, **Label**

```
public class MainActivity : FormsAppCompatActivity {  
    protected override void OnCreate(Bundle bundle)  
    {  
        base.OnCreate(bundle);  
        Forms.SetFlags("FastRenderers_Experimental");  
        global::Xamarin.Forms.Forms.Init(this, bundle);  
        LoadApplication(new FormsApp(pageType));  
        //...  
    }  
}
```



Similmente, sia su Android che iOS è possibile saltare alcuni Layout Cycle del VT, specificando l'attached property **CompressedLayout.IsHeadless**

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
  <StackLayout CompressedLayout.IsHeadless="True">
    <Label Text="Enter a name:" />
  </StackLayout>
</ContentPage>
```

N.B. Eliminano il Renderer dal Layout → NO Gesture, Attached Property ecc.



Demo

Xamarin.Forms Embedding

bit.ly/2z7Z8Cc

Grazie

Domande?



github.com/francedot



[@francedot](https://twitter.com/francedot)



[francesco-bonacci-70428a121](https://www.linkedin.com/in/francesco-bonacci-70428a121)

francesco.bonacci@studentpartner.com



Coffee Break



Lab

App connesse con Xamarin.Forms

bit.ly/2ABbrcu