Chapitre IV

Xamarin.Forms
(Notions de base) (I)

Plan

1. Xamarin.Forms XAML
2. Views (contrôles) et Cellules
3. Views
4. Layouts

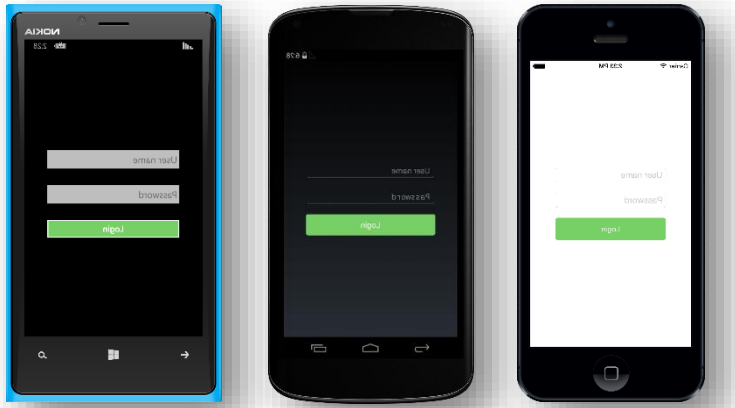
1. Xamarin.Forms XAML

- Deux façons pour construire des interfaces graphiques utilisateur (GUI) en Xamarin
  - En utilisant C# dans le fichier source (.cs)
  - En utilisant XAML dans le fichier d'interface (.xaml)
- **Il est préférable de définir les GUI moyennant XAML**
  - Séparation présentation/traitement
  - Plus facile à comprendre et à maintenir
  - Possibilité d'aperçu (preview) sans exécution complète du code
- eXtensible Application Markup Language
  - Totalement XML

1. Xamarin.Forms XAML

- **XAML : interfaces multi-plateformes**

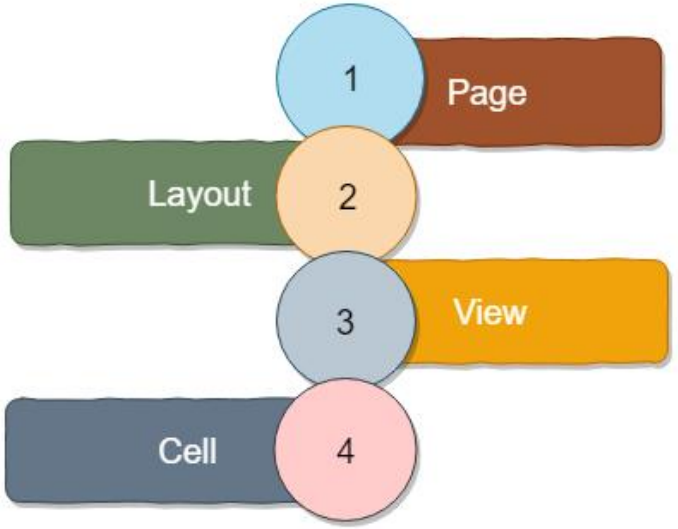
```
<ContentPage>
    <StackLayout Spacing="20" Padding="50"
     VerticalOptions="Center">
        <Entry Placeholder="User name" />
        <Entry Placeholder="Password" IsPassword="True" />
        <Button Text="Login" TextColor="White"
                BackgroundColor="##FF77D065" />
    </StackLayout>
</ContentPage>
```



Xamarin

1. Xamarin.Forms XAML

- Structure d'une interface XAML
  - Page
    - Layout
      - Views (contrôles)
        - cellules



Xamarin

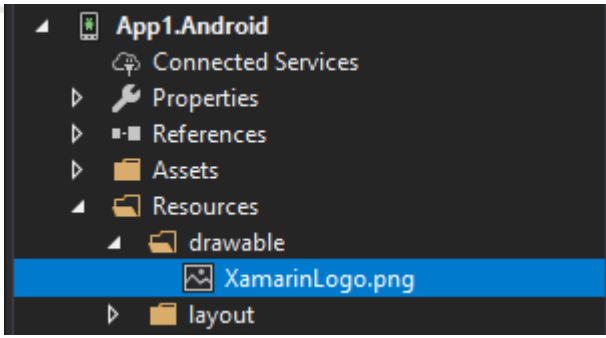
2. Views (contrôles) et Cellules

- Les contrôles sont de type Button, Label, WebView et de cellule (utilisables dans ListView et TableView)

ActivityIndicator	BoxView	Button	DatePicker	Editor
Entry	Image	Label	ListView	Map
OpenGLView	Picker	ProgressBar	SearchBar	Slider
Stepper	Switch	TableView	TimePicker	WebView
EntryCell	ImageCell	SwitchCell	TextCell	ViewCell

2. Views : Image et Label

- Exemple : Image et Label




```
<Image Source="XamarinLogo"
   Aspect="AspectFit"
   HeightRequest="{OnPlatform iOS=300, Android=250}"
   WidthRequest="{OnPlatform iOS=300, Android=250}"
   HorizontalOptions="Center" />
<Label Text="Ceci est une image"
   TextColor="Blue"
   FontAttributes="Italic"
   FontSize="22"
   TextDecorations="Underline"
   HorizontalOptions="Center" />
```

Xamarin

2. Views : WebView

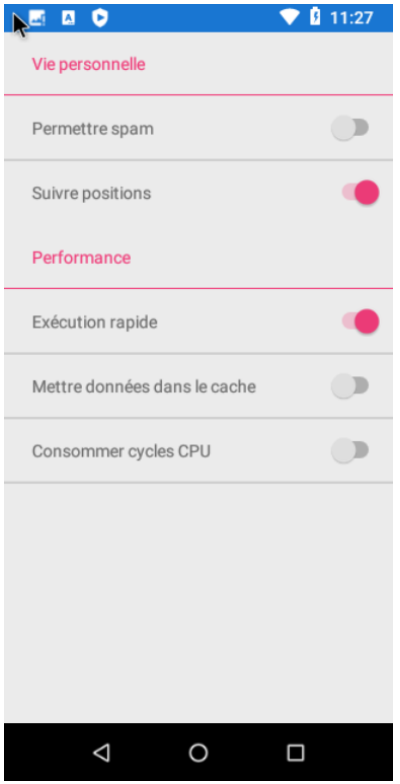
- Exemple : WebView

```
<WebView Source="http://cyberawareness.tn"
         MinimumWidthRequest="200"
         MinimumHeightRequest="200"
         HorizontalOptions="FillAndExpand"
         VerticalOptions="FillAndExpand" />
```

2. Views : TableView et SwitchCell

- Exemple : TableView et SwitchCell

```
<TableView>
  <TableView.Root>
    <TableSection Title="Vie personnelle">
      <SwitchCell Text="Permettre spam" />
      <SwitchCell Text="Suivre positions" On="True" />
    </TableSection>
    <TableSection Title="Performance">
      <SwitchCell Text="Exécution rapide" On="True" />
      <SwitchCell Text="Mettre données dans le cache" />
      <SwitchCell Text="Consommer cycles CPU" />
    </TableSection>
  </TableView.Root>
</TableView>
```

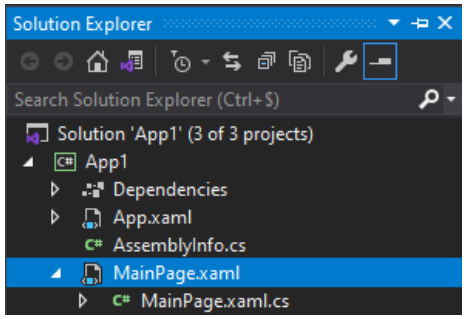


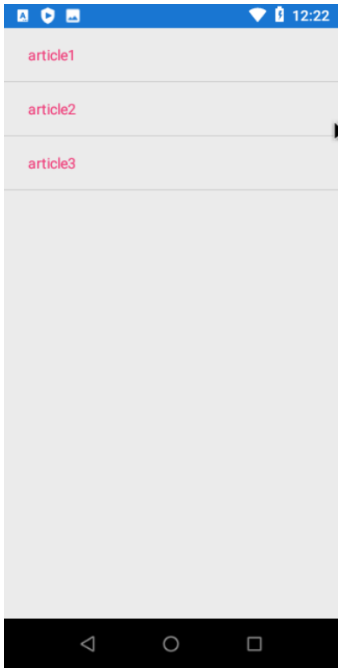
2. Views : ListView simple

- Exemple : ListView simple
  - Dans MainPage.xaml

```
<ListView x:Name="lst"></ListView>
```

  - Dans MainPage.xaml.cs
    - A l'intérieur du constructeur de MainPage






```
public MainPage()
        {
            InitializeComponent();
            lst.ItemsSource = new List<string>() { "article1",
"article2", "article3" };
        }
```

2. Views : ListView avec valeurs attachées (1/2)

- Exemple (1/2) : ListView avec valeurs attachées
  - Dans MainPage.xaml

```
<ListView x:Name="lst">
 <ListView.ItemTemplate>
  <DataTemplate>
   <TextCell Text="{Binding Nom}" Detail="{Binding Num}" >
  </DataTemplate>
 </ListView.ItemTemplate>
</ListView>
```

  - Dans MainPage.xaml.cs
    - A l'intérieur du constructeur de MainPage

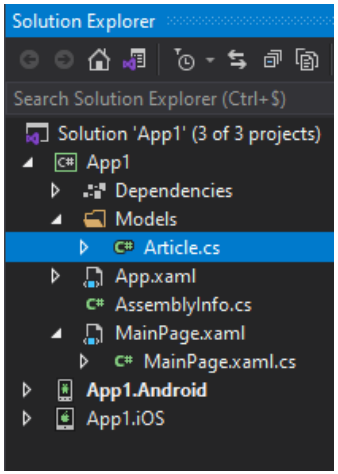
```
public MainPage()
        {
            InitializeComponent();
            lst.ItemsSource = new List<Article>() {
                new Article() {  Nom = "Article1", Num = "0456445450945", },
                new Article() {  Nom = "Article2", Num = "034456445905",  },
                new Article() {  Nom = "Article3", Num = "56445905", },
            };
        }
```

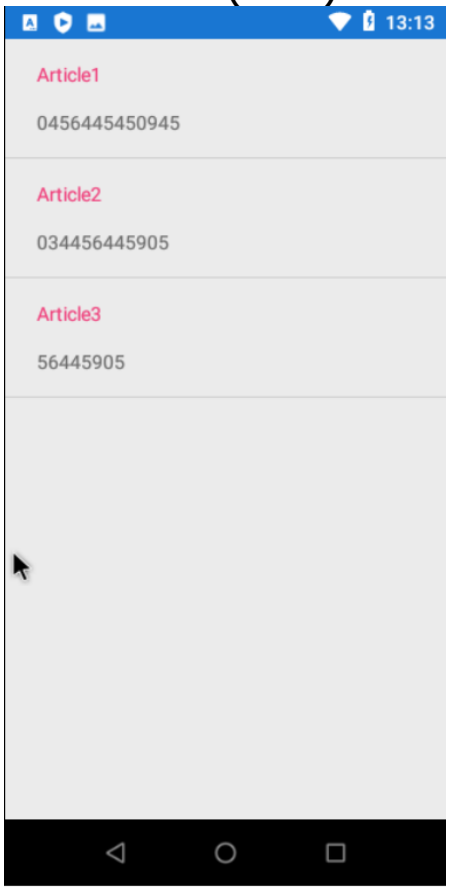
2. Views : ListView avec valeurs attachées (2/2)

- Exemple (2/2) : ListView avec valeurs attachées
  - On ajoute aussi une classe **Article.cs** dans un répertoire **Models** (crée)

```
class Article
    {
            public string Nom { get; set; }
            public string Num { get; set; }

    }
```





2. Views : ListView avec valeurs + images attachées (1/2)

- Exemple (1/2) : ListView avec valeurs + images attachées
  - Dans MainPage.xaml

```
<ListView x:Name="lst">
 <ListView.ItemTemplate>
  <DataTemplate>
   <ImageCell  Text="{Binding Nom}" Detail="{Binding Num}" ImageSource="{Binding
Image}" >
  </DataTemplate>
 </ListView.ItemTemplate>
</ListView>
```

  - Dans MainPage.xaml.cs
    - A l'intérieur du constructeur de MainPage

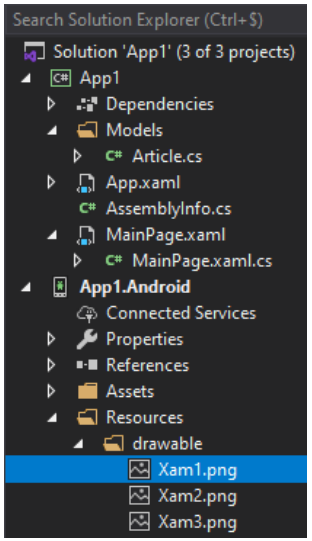
```
public MainPage()
        {
            InitializeComponent();
            lst.ItemsSource = new List<Article>() {
                new Article() {  Nom = "Article1", Num = "0456445450945", Image = "Xam1", },
                new Article() {  Nom = "Article2", Num = "034456445905", Image = "Xam2", },
                new Article() {  Nom = "Article3", Num = "56445905", Image = "Xam3", },
            };
        }
```

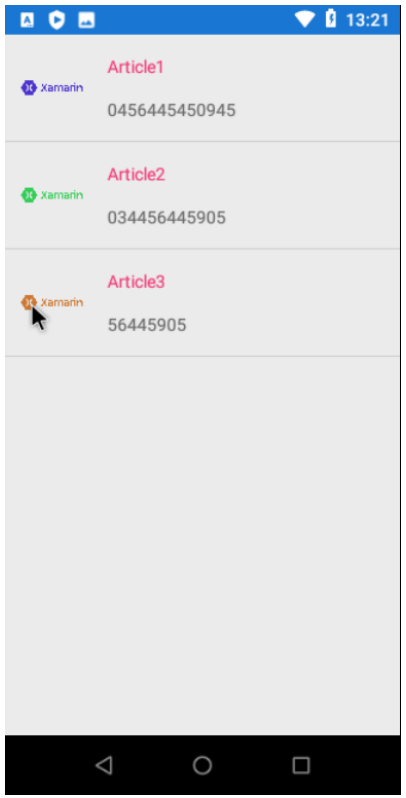
2. Views : ListView avec valeurs + images attachées (2/2)

- Exemple (2/2) : ListView avec valeurs + images attachées
  - On ajoute aussi une classe **Article.cs** dans un répertoire **Models** (crée)

```
class Article
    {
            public string Nom { get; set; }
            public string Num { get; set; }
            public string Image { get; set; }

    }
```

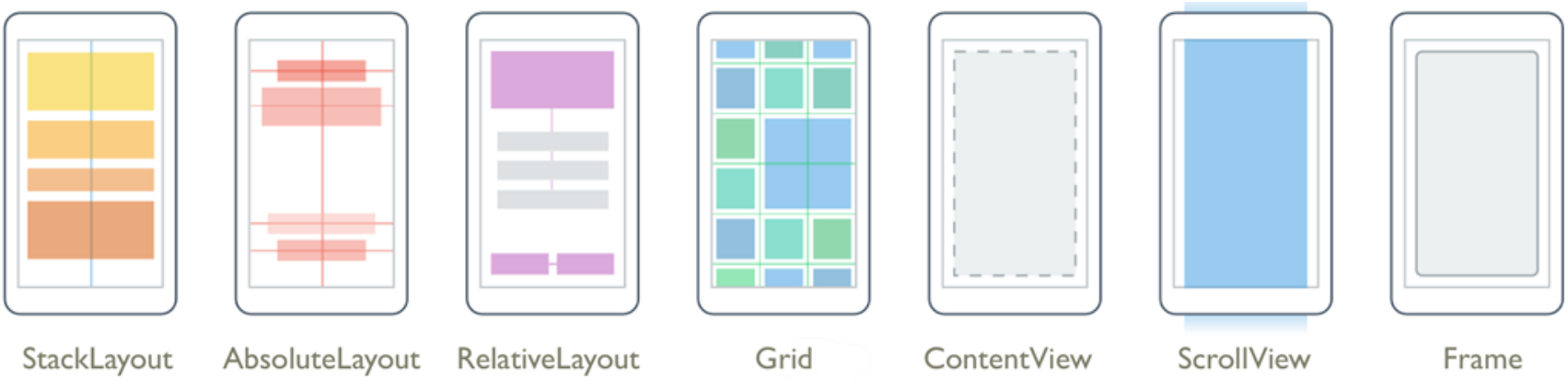




Xamarin

3. Layouts

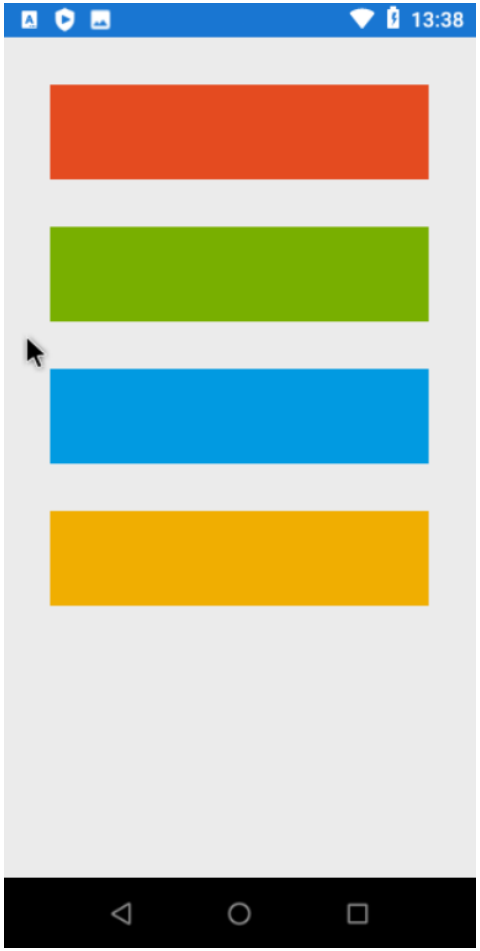
- Layouts?
  - Contrôles pouvant contenir d'autres contrôles!
  - Fournissent :
    - Mise en forme
    - Positionnement



3. Layouts : StackLayout

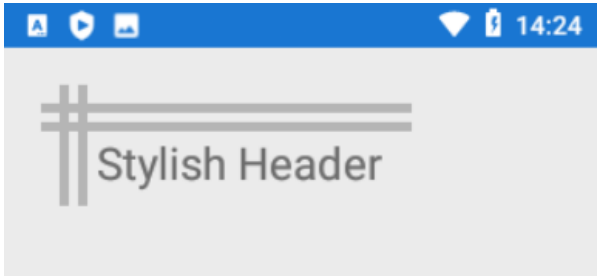
- Organise les contrôles sous forme d'un tas (seule direction)
  - StackOrientation
    - Vertical (défaut) / Horizontal
  - Spacing (double)
    - espace entre contrôles (défaut = 6 device independant units)
- Autre
  - HorizontalOptions
    - Start, Center, End, Fill (+ AndExpand)

```
<StackLayout Padding="32" Spacing="32">
 <BoxView Color="#FFF25022" HeightRequest="64" />
 <BoxView Color="#FF7FBA00" HeightRequest="64" />
 <BoxView Color="#FF01A4EF" HeightRequest="64" />
 <BoxView Color="#FFFFB901" HeightRequest="64" />
</StackLayout>
```



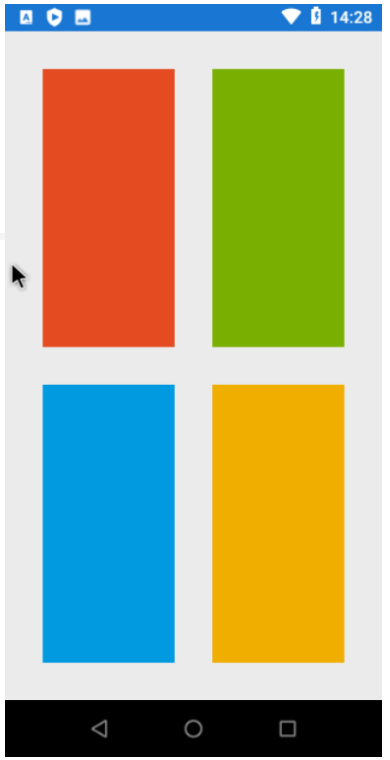
3. Layouts : AbsoluteLayout

- Organise les contrôles avec positionnement direct (position et taille en device-independant units)
  - AbsoluteLayout.LayoutBounds (contrôles)
    - x, y
    - Ou bien x, y, width, height
    - AbsoluteLayout.AutoSize pour le calcul automatique de largeur/hauteur




```
<AbsoluteLayout Margin="20">
 <BoxView Color="Silver" AbsoluteLayout.LayoutBounds="0, 10, 200, 5" />
 <BoxView Color="Silver" AbsoluteLayout.LayoutBounds="0, 20, 200, 5" />
 <BoxView Color="Silver" AbsoluteLayout.LayoutBounds="10, 0, 5, 65" />
 <BoxView Color="Silver" AbsoluteLayout.LayoutBounds="20, 0, 5, 65" />
 <Label Text="Stylish Header" FontSize="24"
        AbsoluteLayout.LayoutBounds="30, 25" />
</AbsoluteLayout>
```

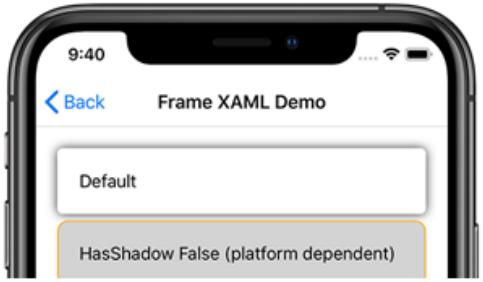
3. Layouts : GridLayout

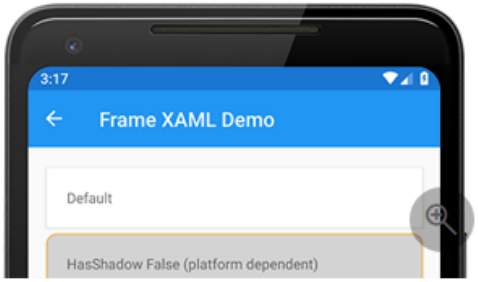



```
<Grid Padding="32" RowSpacing="32" ColumnSpacing="32">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <BoxView Grid.Row="0" Grid.Column="0" Color="#FFF25022" />
        <BoxView Grid.Row="0" Grid.Column="1" Color="#FF7FBA00" />
        <BoxView Grid.Row="1" Grid.Column="0" Color="#FF01A4EF" />
        <BoxView Grid.Row="1" Grid.Column="1" Color="#FFFFB901" />
</Grid>
```

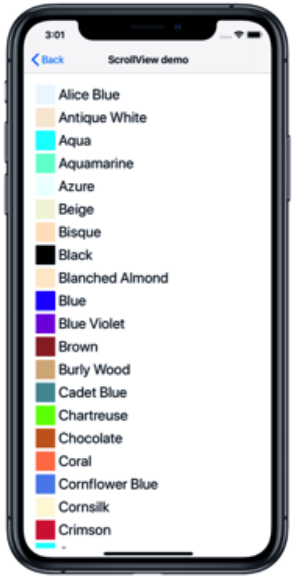
3. Layouts additionnels : Frame, ScrollView, et ContentView

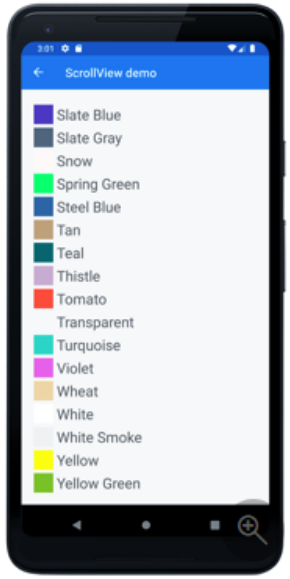
Frame





ScrollView





ContentView