ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
Σχολή Επιστημών Πληροφορίας
Τμήμα Εφαρμοσμένης Πληροφορικής

Επιβλέπων Καθηγητής:
ΓΕΩΡΓΙΑΔΗΣ ΧΡΗΣΤΟΣ
(geor (at) uom.gr)

Συγγραφή:
ΚΑΛΑΪΤΖΙΔΗΣ ΑΝΑΣΤΑΣΙΟΣ
(mai17042@uom.edu.gr)

Συνοδευτικό υλικό:
- Project στοιχείων Xamarin.Forms
- Project δημιουργίας e-shop
- Σειρά εγχειριδίων
- Γλωσσάρι
- Δημοφιλείς ιδιότητες στοιχείων
- Παραπομπές

Διαθέσιμο στο github:
https://github.com/iwizu/LearnXamarinForms

Xamarin.Forms

Project : "Δημιουργία Cross-platform ηλεκτρονικού καταστήματος Βιβλίων"

One C# code, multiple platforms


Android


iOS


Windows Phone


Windows 10

1. Παρουσίαση του Xamarin.Forms: Εγκατάσταση, XAML, Pages, Layouts

1. Παρουσίαση του Xamarin.Forms: Εγκατάσταση, XAML, Pages, Layouts

1.1. Η επανάσταση του Xamarin.Forms

Η εταιρία Xamarin ιδρύθηκε το 2011 από τους μηχανικούς του project mono, με σκοπό τη δημιουργία εργαλείων που θα επέτρεπαν την ανάπτυξη εφαρμογών που εκτελούνται σε πολλές πλατφόρμες, όπως Android, IOS και Windows με διαμοιρασμό κώδικα της γλώσσας C#.

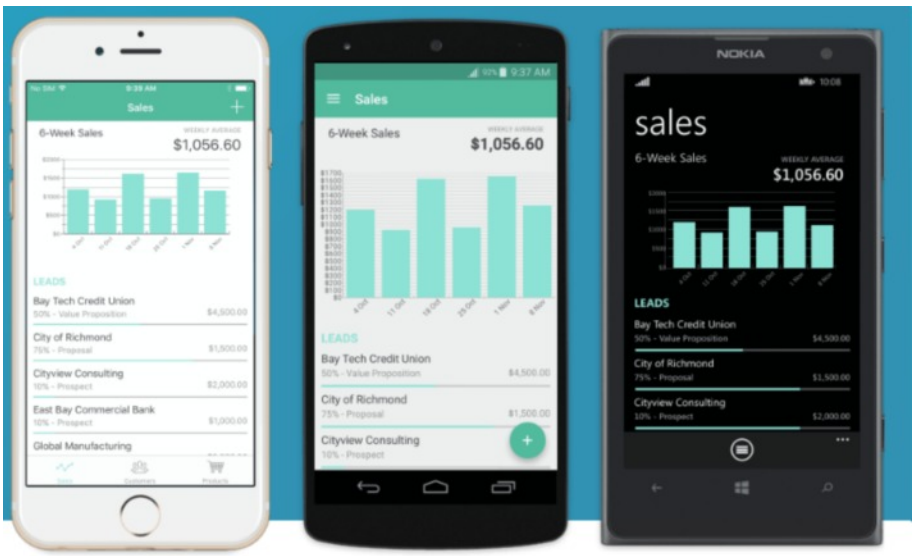
Πλέον, οι χρήστες δεν απαιτείται να μάθουν πολλές τεχνολογίες για να αναπτύξουν την ίδια εφαρμογή σε διαφορετικές πλατφόρμες. Για παράδειγμα δεν απαιτείται η εκμάθηση:

- Java για να αναπτύσσουν εφαρμογές Android στο Android Studio (ή στο Eclipse),
- Objective C ή SWIFT για να αναπτύσσουν εφαρμογές iOS
- WinForms, WPF ή Universal Windows Apps για να αναπτύσσουν εφαρμογές Windows 10/Phone.

Με τη γνώση της C#, της περιγραφικής γλώσσας XAML και του IDE Visual Studio γράφουν μια φορά τον κώδικα και τον διαμοιράζουν σε όλες τις πλατφόρμες. Σχεδόν το 95% του κώδικα μπορεί να διαμοιραστεί με τη χρήση του Xamarin.Forms.

Η εταιρία αρχικά ανέπτυξε τις εκδόσεις Xamarin.Android και Xamarin.iOS που επιτρέπουν το χρήστη να αναπτύσσει εφαρμογές αποκλειστικά για τις εν λόγω πλατφόρμες με κώδικα C#. Έτσι για παράδειγμα το Xamarin.Android επιτρέπει το χρήστη να εκμεταλλευτεί όλες τις δυνατότητες (controls, layouts, sensors κτλ) της βιβλιοθήκης του Android SDK.

Αργότερα προέκυψε το Xamarin.Forms που πλέον επιτρέπει την ανάπτυξη ενός κώδικα-μιας βιβλιοθήκης, που μετασχηματίζεται σε πολλές πλατφόρμες. Έτσι για παράδειγμα χρησιμοποιούμε τα ίδια layouts, controls κτλ τα οποία μέσα από τα εργαλεία του Xamarin μετασχηματίζονται σε στοιχεία που υποστηρίζει το Android και το iOS.



Εικόνα 1: Xamarin.Forms-Μια εφαρμογή μετασχηματίζεται σε πολλές πλατφόρμες (src: xamarin.com)

1.2. Το project "UniBook: A Cross-platform ebook store"

Στην παρούσα σειρά εγχειριδίων θα επιχειρήσουμε να αναπτύξουμε σταδιακά τη γνώση μας, ξεκινώντας από τις βασικές έννοιες σχεδίασης εφαρμογών κινητού. Τελικός στόχος είναι βήμα προς βήμα να δημιουργήσουμε μία εφαρμογή κινητού για πολλές πλατφόρμες, η οποία θα αφορά ένα ηλεκτρονικό κατάστημα ψηφιακών βιβλίων, το UniBook.

Θα ξεκινήσουμε από μια πρόχειρη σχεδίαση του layout, της πλοήγησης και βασικών controls που θα χρησιμοποιηθούν.

Έπειτα θα μελετήσουμε τον τρόπο αποθήκευσης δεδομένων μέσα σε μια Βάση Δεδομένων SQLite και τη λήψη δεδομένων με την αξιοποίηση δικτυακών API, όπως το GoodReads API.

Τέλος, θα μάθουμε για τη σύνδεση σε υπηρεσίες cloud, όπως το Microsoft Azure, η διασύνδεση δεδομένων, το animation και την τελική δημοσίευση της εφαρμογής μας.

Θα παρατηρήσετε ότι στα εγχειρίδια περιπλέκεται η Ελληνική γλώσσα με αγγλικούς όρους. Αυτό έγινε σκόπιμα, προκειμένου ο αναγνώστης να εξοικειωθεί με τους τεχνικούς όρους που θα συναντάει διαρκώς στο διαδίκτυο. Για τυχόν απορίες ανατρέξτε στο γλωσσάρι, όπου θα βρείτε τη μετάφρασή τους.

Tags: #Xamarin.Forms, #project_UniBook

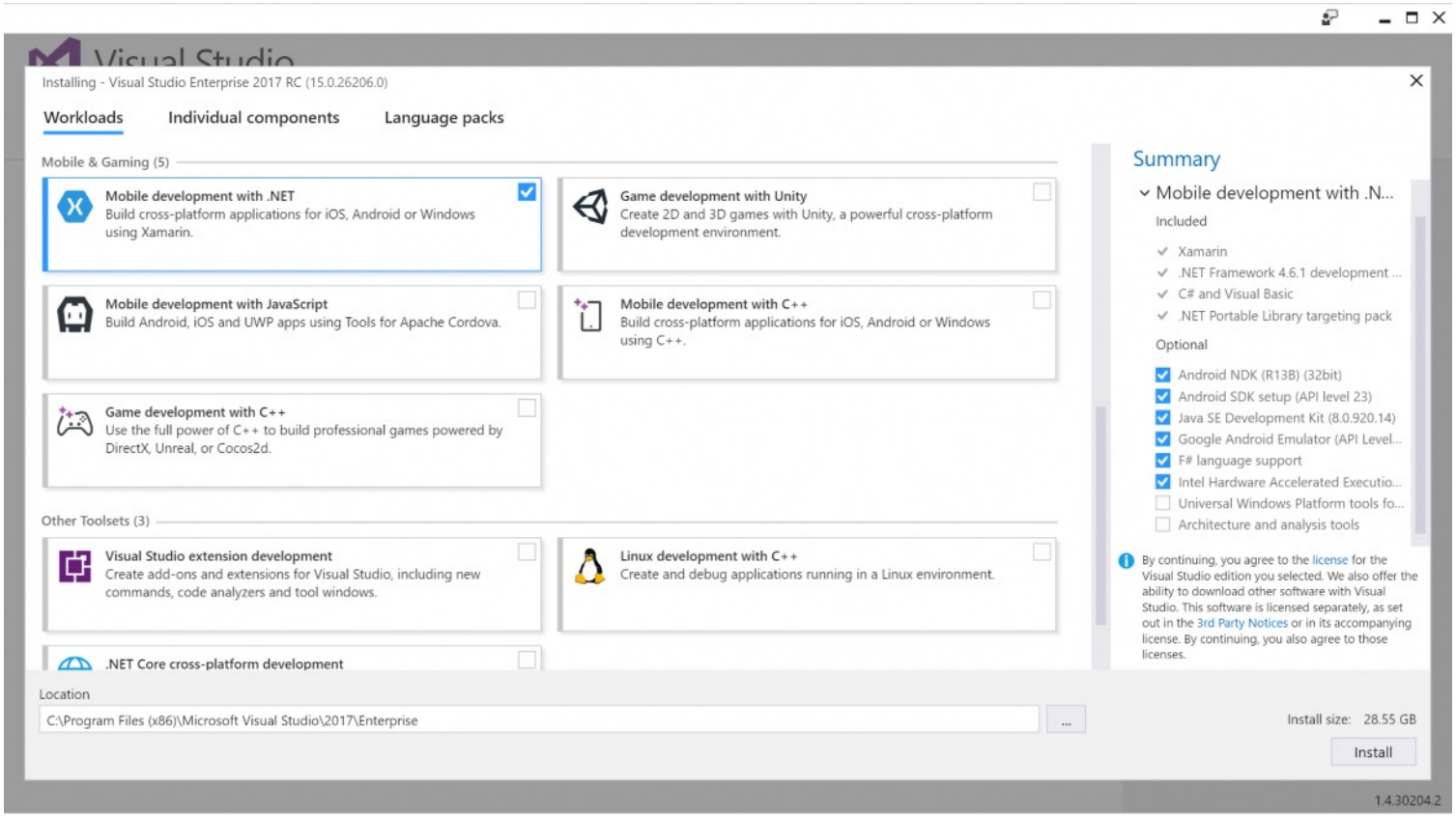
1.3. Προαπαιτούμενα

Στα πλαίσια των οδηγών πρέπει να εγκαταστήσουμε τα παρακάτω λογισμικά:

- **Pencil**: Ένα αξιόλογο εργαλείο προτυποποίησης, στο οποίο θα σχεδιάσουμε το User Interface της εφαρμογής μας προτού πιάσουμε κώδικα. Θα το βρείτε εδώ: http://pencil.evolus.vn/
- **Visual Studio 2017 Community edition**: Ένα από τα πιο δημοφιλή περιβάλλοντα IDE για την ανάπτυξη εφαρμογών desktop, web και mobile. Περιλαμβάνει τα εργαλεία του Xamarin και θα το βρείτε εδώ: https://www.visualstudio.com/downloads/

1.4. Εγκατάσταση

Αφού μεταφορτώσετε το Visual Studio 2017 Community edition, το εκτελείτε και απλά επιλέγετε τις ενότητες που σας ενδιαφέρουν. Για να συμπεριλάβετε το Xamarin επιλέξτε το "Mobile development with .NET".
Η διαδικασία εγκατάστασης είναι πολύ απλή, όμως αν χρειαστείτε βοήθεια, εδώ θα βρείτε έναν πολύ χρήσιμο οδηγό: https://www.youtube.com/watch?v=BDdjG--LmhE



Εικόνα 2: Εγκατάσταση Visual Studio 2017 Community edition

1.5. XAML

Η σχεδίαση του User Interface μπορεί να γίνει είτε με τη γλώσσα C#, είτε με την περιγραφική γλώσσα XAML είτε με συνδυασμό των δύο. Στο project θα χρησιμοποιήσουμε και τις δύο γλώσσες, τη XAML και λιγότερο τη C# για το επίπεδο του User Interface και τη C# για τα επίπεδα της business logic και του data access.
Όσοι δεν γνωρίζουν τη C# μπορούν να βρουν ένα καταπληκτικό βιβλίο εδώ: http://www.csharpcourse.com/
Η XAML είναι μια πανίσχυρη περιγραφική γλώσσα με την οποία μπορούμε να σχεδιάσουμε το γραφικό περιβάλλον της εφαρμογής. Ουσιαστικά πρόκειται για ένα xml αρχείο με κατάληξη .xaml. Το αρχείο ξεκινάει με τον εξής κώδικα:

```
<?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="XamlSamples.HelloXamlPage">
</ContentPage>
```

Tags: #Προαπαιτούμενα, #Εγκατάσταση, #XAML

Αυτό που χρειάζεται να γνωρίζουμε είναι το x:Class, που δείχνει το Namespace και την κλάση που δημιουργείται από αυτό το XAML αρχείο. Σε αυτή την κλάση μπορούμε να γράψουμε κώδικα C# και να υλοποιήσουμε το business process. Σε περίπτωση που χρειαστεί να δηλώσουμε ένα νέο Namespace απλά προσθέτουμε:

xmlns:local="clr-namespace:UniBook"

Ας δούμε πως μπορούμε να δημιουργήσουμε μια ετικέτα κειμένου με δύο διαφορετικούς τρόπους: τη XAML και τη C#.

XAML-MainPage.xaml

```
<Label Text="Welcome to Xamarin Forms!"
       VerticalOptions="Center"
       HorizontalOptions="Center" />
```

C#-MainPage.xaml.cs

```
Label label = new Label();
 label.Text = "Welcome to Xamarin Forms!";
label.VerticalOptions = LayoutOptions.Center;
label.HorizontalOptions = LayoutOptions.Center;
```

Με το <Label δηλώνουμε ότι επιθυμούμε μια ετικέτα κειμένου. Στο τέλος κλείνουμε τη δήλωση με το /> . Ενδιάμεσα εισάγουμε τις ιδιότητες με τη μορφή Ιδιότητα=Τιμή . Έτσι με το *Text=”Welcome”* δηλώνουμε ότι το κείμενο της ετικέτας είναι Welcome. Το *VerticalOptions=”Center”* δηλώνει Κάθετη Στοίχιση στο κέντρο, ενώ το *HorizontalOptions=”Center”* δηλώνει κατακόρυφη στοίχιση στο κέντρο. Άλλες γνωστές ιδιότητες της ετικέτας κειμένου είναι οι εξής:

- HorizontalTextAlignment="Center"
- Rotation="-15"
- IsVisible="true"
- FontSize="Large"
- FontAttributes="Bold"
- TextColor="Aqua"

Παρατηρούμε ότι ενώ στη γλώσσα C# γράψαμε VerticalOptions=LayoutOptions.Center στη XAML γράφουμε VerticalOptions=”Center”. Παραδείγματα αποκοπής επιπρόσθετων λέξεων, όπως το LayoutOptions συμβαίνουν συχνά στη XAML και μας διευκολύνουν ώστε να γράψουμε πιο γρήγορα την τιμή της ιδιότητας. Οι κλάσεις που περιλαμβάνουν την ιδιότητα αυτή προκύπτουν από την υπερκλάση TypeConverters.

1.6. Pages

Η κλάση Page είναι ένα οπτικό αντικείμενο που απασχολεί το μεγαλύτερο μέρος της οθόνης και περιέχει ένα μόνο παιδί. Η Xamarin.forms διαθέτει τα παρακάτω Pages:

- ContentPage: Αποτελεί μια μόνο όψη
- MasterDetailPage: Προβάλει δύο όψεις, μία Master και μία Detail
- NavigationPage: Χειρίζεται αποτελεσματικά την πλοήγηση μεταξύ σελίδων
- TabbedPage: Επιτρέπει την πλοήγηση σε μεταξύ θυγατρικών σελίδων με τη χρήση tabs
- TemplatedPage: Προβάλει περιεχόμενο σε ολόκληρη την οθόνη
- CarouselPage: Επιτρέπει την αλλαγή σελίδων με τη μορφή gallery που κινούμε σέρνοντας το δάχτυλό μας

Tags: #Label, #Pages

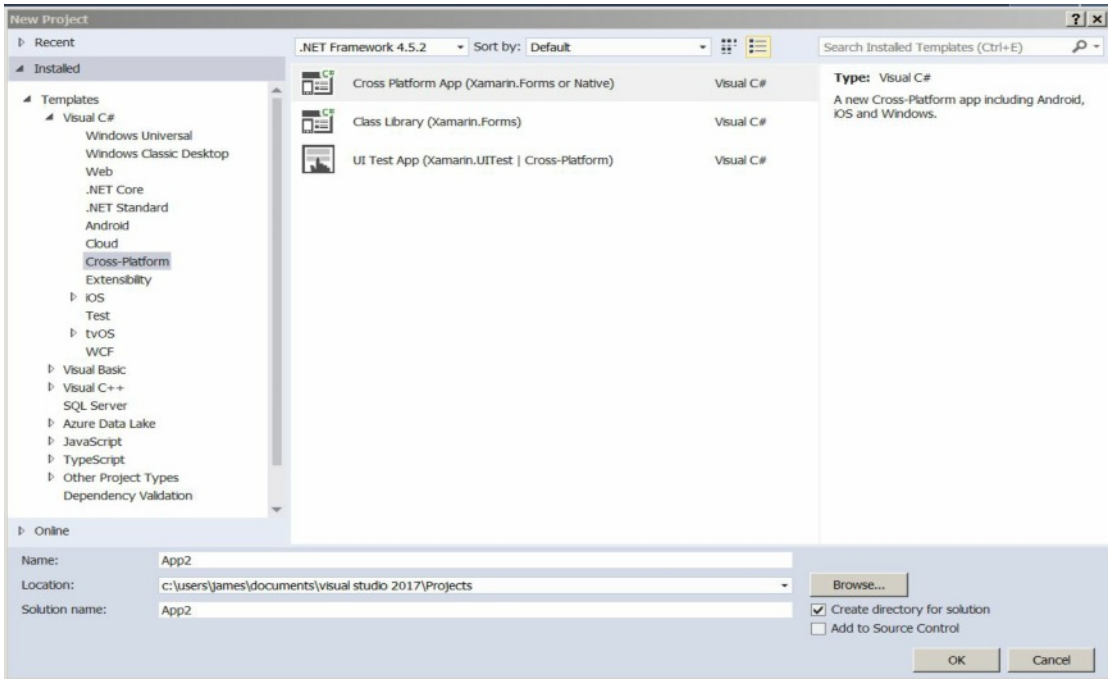
1.7. Τα βοηθητικά projects

Στα πλαίσια του παρόντος μαθήματος αναπτύχθηκαν δύο συνοδευτικά projects. Το ένα περιέχει τα τεχνικά στοιχεία που μαθαίνουμε σε κάθε εγχειρίδιο, ενώ το δεύτερο περιέχει το project του ηλεκτρονικού καταστήματος που αναπτύσσουμε.
Θα βρείτε τα project στο github και συγκεκριμένα στη διεύθυνση:
https://github.com/iwizu/LearnXamarinForms
Κάθε φάκελος περιέχει την αρίθμηση του εκάστοτε μαθήματος που αφορά.

1.8. Ξεκινώντας ένα νέο project

Για να δημιουργήσουμε μια νέα εφαρμογή, αφού έχουμε εγκαταστήσει το Visual Studio 2017 με υποστήριξη Xamarin, το ανοίγουμε και εκτελούμε τα εξής βήματα:

- Επιλέγουμε File→New Project
- Έπειτα επιλέγουμε από την αριστερή λίστα Cross-platform και από τη δεξιά λίστα Cross-platform App δίνοντας το όνομα της εφαρμογής στο πεδίο “Name:”.



- Έπειτα στο παράθυρο που θα εμφανιστεί επιλέγουμε “Blank App” και ως Code Sharing Strategy το Portable Class Library (PCL) που είναι και ο πιο συχνός τρόπος διαμοιρασμού κώδικα.

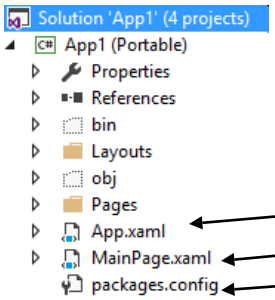
Η διαφορά τους είναι η εξής: Με το Shared project οργανώνουμε τον κώδικα χρησιμοποιώντας #if directives για να διαφοροποιήσουμε τον κώδικα ανά πλατφόρμα. Με το Portable Class Library (PCL) χρησιμοποιούμε Interfaces για να διαφοροποιήσουμε τον κώδικα ανάλογα με την κάθε πλατφόρμα. Για περισσότερες πληροφορίες δείτε εδώ: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/code-sharing/ .
Αφού φορτώσει το project μας, παρατηρούμε στο δεξί μέρος το Solution explorer. Περιέχει όλη τη δομή των αρχείων του project με τέσσερις ενότητες:

- App1 (Portable): Περιέχει τον κοινό κώδικα
- App1.Android: Ο κώδικας που θα χρησιμοποιηθεί στο Android

Tags: #github, #New_Project

- App.iOS: Ο κώδικας που θα χρησιμοποιηθεί στο iOS
- App.UWP: Ο κώδικας που θα χρησιμοποιηθεί στο Universal Windows Apps

Η δομή του project είναι η παρακάτω:



1.Το πρώτο αρχείο που φορτώνεται στην εφαρμογή
2.Το App φορτώνει την πρώτη σελίδα MainPage
Φιλοξενεί τα πρόσθετα πακέτα της εφαρμογής

Ανοίγουμε το MainPage κάνοντας διπλό κλικ και γράφουμε τον παρακάτω κώδικα:

```
<Label Text="Welcome to Xamarin Forms!"
       VerticalOptions="Center"
       HorizontalOptions="Center" />
  <Label Text="---PAGES---"
       VerticalOptions="Start"
       HorizontalOptions="StartAndExpand" />
  <Button Text="Pages" Clicked="Button_Clicked"
        VerticalOptions="Start"
        HorizontalOptions="StartAndExpand" />
```

Στην αρχή τοποθετούμε δύο ετικέτες κειμένου και μετά ένα κουμπί Button. Στο κουμπί θέτουμε την ιδιότητα κειμένου ως “Pages”. Η ιδιότητα Clicked παίρνει ως τιμή το όνομα της μεθόδου που χειρίζεται το συμβάν Πάτημα κουμπιού. Έτσι αν κάνουμε κλικ στο κουμπί θα εκτελεστεί η μέθοδος Button_Clicked:

```
private void Button_Clicked(object sender, EventArgs e)
      {
        Navigation.PushAsync(new AContentPage());
      }
```

Η μέθοδος μας πλοηγεί σε μια νέα σελίδα. Παρακάτω θα μάθουμε περισσότερα για την πλοήγηση στο περιβάλλον της Xamarin.Forms.

1.9. Πλοήγηση στις σελίδες

Σπάνια έχουμε μια εφαρμογή που απασχολεί μία μόνο οθόνη. Συνήθως οι εφαρμογές περιέχουν πολλές οθόνες και ένα σύστημα πλοήγησης μεταξύ τους. Όπως είδαμε το App.xaml είναι το πρώτο στοιχείο που εκτελείται κατά την έναρξη της εφαρμογής. Ο πιο απλός τρόπος για να μεταβούμε από το App.xaml στη MainPage.xaml (ή όποια άλλη σελίδα επιθυμούμε) είναι ο παρακάτω κώδικας στον κατασκευαστή του App, αμέσως μετά τη γραμμή InitializeComponent(); :

```
MainPage = new NavigationPage(new MainPage());
```

Το NavigationPage είναι μια κλάση που χειρίζεται την πλοήγηση μεταξύ παραθύρων. Κάθε φορά που καλείται για να μας ανακατευθύνει σε μια νέα σελίδα, τοποθετεί σε στοίβα όλο το ιστορικό κλήσεων και μας επιτρέπει από κάθε σημείο να επιστρέψουμε πίσω στις προηγούμενες. Από τη MainPage και έπειτα μπορούμε να συνεχίσουμε τη μετάβαση σε άλλες σελίδες με την πιο απλή εντολή :

```
Navigation.PushAsync(new AContentPage()); //Δηλαδή βάλε στη στοίβα μια νέα κλάση της σελίδας AContentPage
```

1.10. Page: Η συνέχεια

Έχοντας ανοίξει ένα νέο project και εκτελώντας πλέον πραγματικό κώδικα μπορούμε να δούμε στην πράξη τις διάφορες κλάσεις που υλοποιούν την κλάση Page.

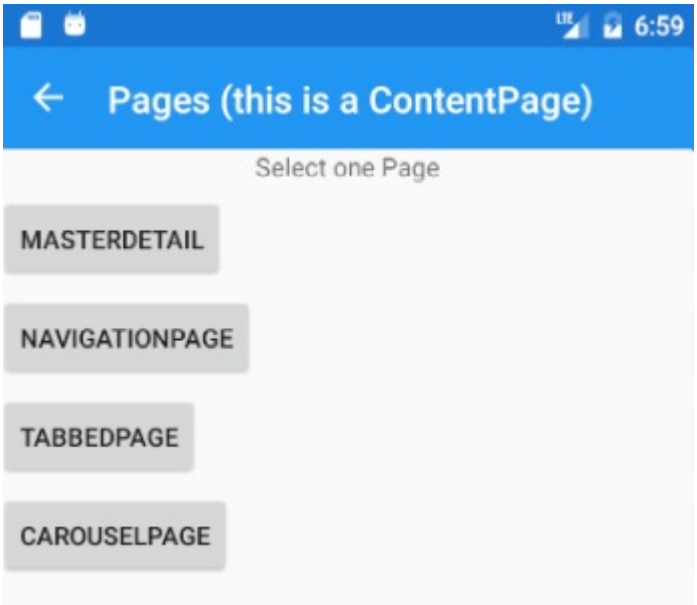
1.10.1. ContentPage

Η ContentPage προβάλει μία μόνο όψη (View), η οποία συνήθως είναι ένα StackLayout ή ένα ScrollView. Στο project μας θα χρησιμοποιήσουμε ένα ContentPage για να φτιάξουμε ένα μενού επιλογών μετάβασης σε άλλες κλάσεις τύπου Page.

- Κάντε δεξί κλικ στο project App και επιλέξτε “Add new folder” και προσθέστε ένα φάκελο με το όνομα Pages. Μέσα στο φάκελο θα προσθέσουμε όλες τις σελίδες που υλοποιούν την κλάση Page.
- Κάντε δεξί κλικ στο φάκελο Pages και επιλέξτε “Add new item”. Στο παράθυρο που θα προβληθεί επιλέξτε “Forms Blank Content Page Xaml” και δώστε το όνομα “AContentPage”. Εισάγετε στη xaml τον κώδικα:

```
<StackLayout>
<Label Text="Select one Page"
      VerticalOptions="Center"
      HorizontalOptions="Center" />
<Button Text="MasterDetail" Clicked="Button_Clicked"
       VerticalOptions="Start"
       HorizontalOptions="StartAndExpand" />
<Button Text="NavigationPage" Clicked="Button_Clicked_1"
       VerticalOptions="Center"
       HorizontalOptions="StartAndExpand" />
<Button Text="TabbedPage" Clicked="Button_Clicked_2"
       VerticalOptions="Center"
       HorizontalOptions="StartAndExpand" />
  <Button Text="CarouselPage" Clicked="Button_Clicked_3"
       VerticalOptions="Center"
       HorizontalOptions="StartAndExpand" />
</StackLayout>
```

Το StackLayout είναι μια δομή οργάνωσης στοιχείων (layout) που στοιβάζει τα αντικείμενα που περιέχει. Είναι κοινό να τοποθετούμε μέσα σε ένα ContentPage ένα μόνο στοιχείο StackLayout.
Πρώτα θέτουμε μία ετικέτα κειμένου και τη στοιχίζουμε στο κέντρο με την ιδιότητα VerticalOptions. Ακολούθως τοποθετούμε τέσσερα κουμπιά Button κάθε ένα από τα οποία οδηγεί σε διαφορετικό τύπο σελίδας.
Για την κάθετη τοποθέτηση χρησιμοποιούμε το VerticalOptions, το οποίο μπορεί να πάρει τιμές όπως Start, Center, End και Fill.

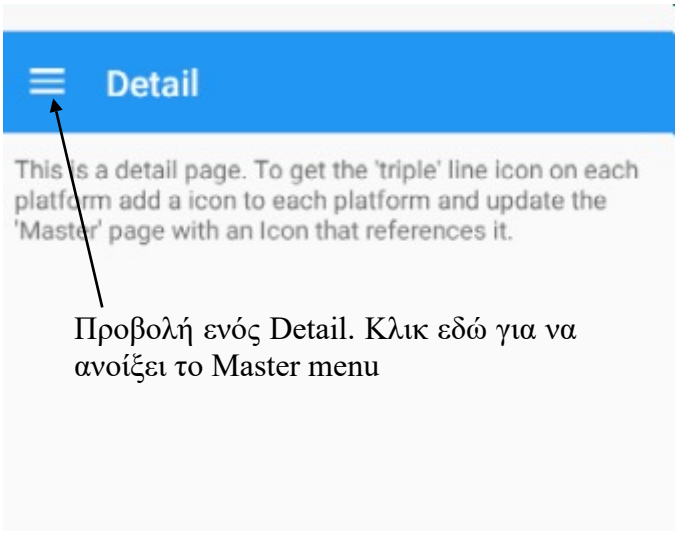


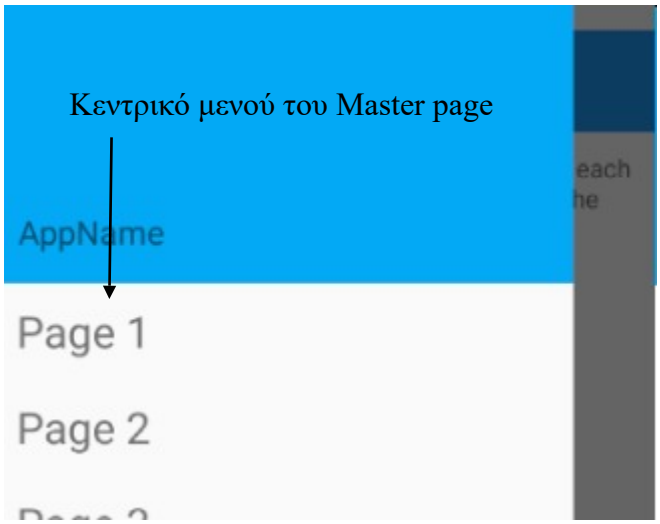
Tags: #ContentPage

1.10.2. MasterDetailPage

Το MasterDetailPage είναι μια κλάση που χρησιμοποιείται για τη διαχείριση δύο ειδών πληροφοριών: μια σελίδα Master που ενσωματώνει δεδομένα σε υψηλό επίπεδο (όπως μενού) και μια σελίδα Detail που προβάλει χαμηλού επιπέδου πληροφορίες σχετικά με το επιλεγμένο στοιχείο του Master.

- Για να δημιουργήσουμε ένα MasterDetailPage κάνουμε δεξί κλικ μέσα στο φάκελο Pages και επιλέγουμε Add new item.
- Έπειτα επιλέγουμε "Forms MasterDetail Page Xaml" και το Visual Studio δημιουργεί για εμάς όλα τα απαραίτητα αρχεία ενός MasterDetailPage.





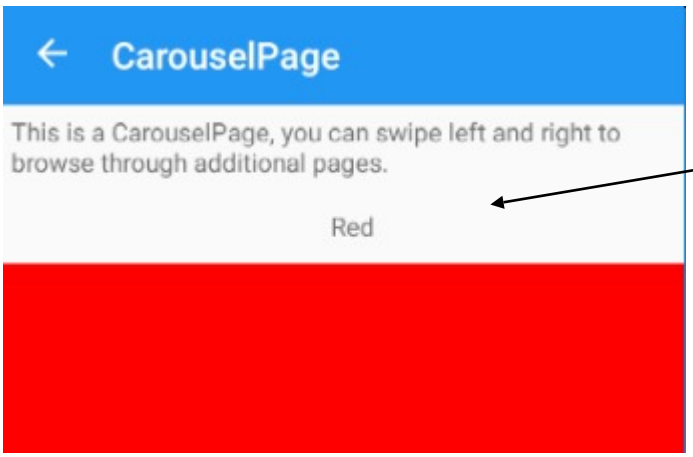
1.10.3. TabbedPage

Το TabbedPage επιτρέπει την πλοήγηση σε διαφορετικά πάνελ πληροφοριών χρησιμοποιώντας tabs. Η πλοήγηση μπορεί να γίνει και σέρνοντας το δάχτυλό μας αριστερά ή δεξιά.
Για να δημιουργήσουμε ένα νέο TabbedPage ακολουθούμε τη γνωστή διαδικασία προσθήκης νέου αντικειμένου και έπειτα επιλέγουμε από τη λίστα προτύπων το "Forms Tabbed Page Xaml".

Tags: #MasterDetailPage, #TabbedPage

1.10.4. CarouselPage

Το CarouselPage είναι μία σελίδα που επιτρέπει την εναλλαγή υποσελίδων με ένα απλό σύρσιμο του δαχτύλου μας. Μοιάζει με μια συλλογή φωτογραφιών που προβάλλουμε στο κινητό μας.
Για να προσθέσουμε ένα CarouselPage χρησιμοποιούμε τη γνωστή διαδικασία προσθήκης, επιλέγοντας “Forms Carousel Page Xaml”.



Απλά σέρνουμε το δάχτυλο αριστερά ή δεξιά για να αλλάξουμε υποσελίδα

1.11. Layouts

Τα Layouts χρησιμοποιούνται για να οργανώσουν τα Controls σε λογικές δομές που βοηθούν το χρήστη να τα αξιοποιήσει παραγωγικά. Τα πιο γνωστά Layouts είναι τα εξής:

- StackLayout : Τοποθετεί τα αντικείμενα που περιέχει σε μια γραμμή, οριζόντια ή κατακόρυφα
- ContentView : Περιέχει ένα μόνο στοιχείο και χρησιμοποιείται κυρίως για ορισμένα από το χρήστη Controls
- Frame : Προβάλει ένα μόνο αντικείμενο με κάποια στοιχεία πλαισίωσης (πχ σκιά)
- ScrollView : Επιτρέπει τις μπάρες ολίσθησης αν το περιεχόμενο είναι πολύ μεγάλο
- TemplatedView : Προβάλει περιεχόμενα ενός πρότυπου Control
- AbsoluteLayout : Τοποθετεί τα περιεχόμενα σε ακριβείς θέσεις. Η θέση και το μέγεθος των Controls καθορίζεται από το χρήστη
- Grid : Τοποθετεί τα περιεχόμενα σε γραμμές και στήλες
- RelativeLayout : Χρησιμοποιεί περιορισμούς για να τοποθετήσει τα περιεχόμενα
- ContentPresenter : Είναι διαχειριστής περιεχομένου που χρησιμοποιείται για προτυποποιημένες όψεις.

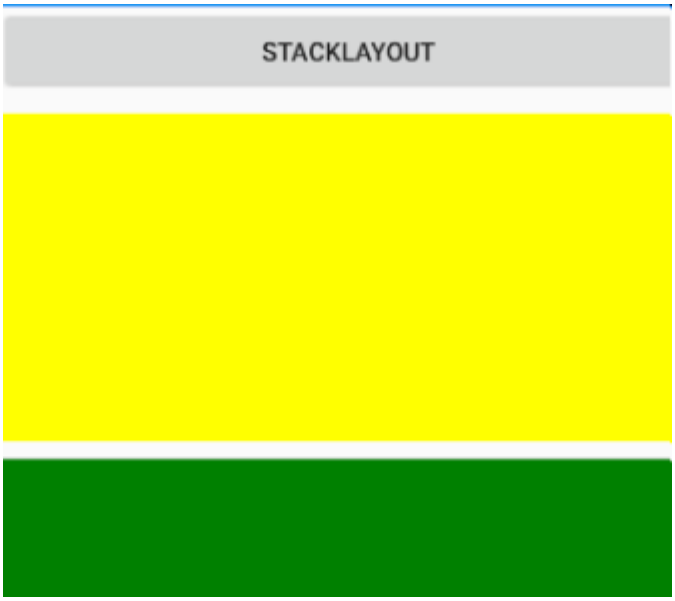
Στο project μας δημιουργήστε ένα φάκελο με το όνομα Layouts. Για κάθε ένα από τα παραδείγματα που θα ακολουθήσουν δημιουργήστε μέσα στο φάκελο ένα αρχείο xaml τύπου ContentPage.

1.11.1. StackLayout

Το StackLayout τοποθετεί τα στοιχεία σε μια γραμμή, οριζόντια ή κατακόρυφα. Ο κώδικας που θα εισάγουμε στο νέο ContentPage που θα δημιουργήσουμε είναι ο εξής:

```
<StackLayout Spacing="10" x:Name="layout">
        <Button Text="StackLayout" VerticalOptions="Start"
    HorizontalOptions="FillAndExpand" />
        <BoxView Color="Yellow" VerticalOptions="FillAndExpand"
    HorizontalOptions="FillAndExpand" />
        <BoxView Color="Green" VerticalOptions="FillAndExpand"
    HorizontalOptions="FillAndExpand" />
        <BoxView HeightRequest="75" Color="Blue" VerticalOptions="End"
    HorizontalOptions="FillAndExpand" />
    </StackLayout>
```

Το StackLayout δηλώνει τη στοίχιση των περιεχομένων σε μια γραμμή. Τα αντικείμενα θα έχουν απόσταση 10 (Spacing=”10”). Με την ιδιότητα x:Name=”layout” δηλώνουμε το όνομα του στιγμιότυπου, το οποίο μπορούμε να χρησιμοποιήσουμε στη C# για να καλέσουμε τις ιδιότητές του.
Έπειτα τοποθετούμε ένα κουμπί στην έναρξη κάτι που δηλώνεται με την ιδιότητα VerticalOptions=”Start”. Το HorizontalOptions=”FillAndExpand” σημαίνει ότι θέλουμε να γεμίσει σε πλάτος και να καλύψει τα όρια του StackLayout.
Τα υπόλοιπα αντικείμενα του StackLayout είναι τύπου BoxView τα οποία είναι απλά κουτιά που περιέχουν ένα χρώμα (δηλώνεται με το Color=”Yellow” κτλ). Βάζοντας σε κάθε ένα το VerticalOptions=”FillAndExpand” δηλώνουμε ότι επιθυμούμε ισόποσα να γεμίσουν οριζόντια τον υπόλοιπο χώρο του StackLayout.



1.11.2. AbsoluteLayout

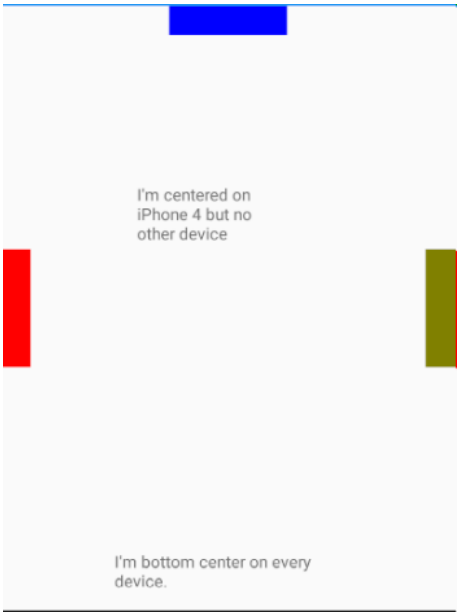
Το AbsoluteLayout τοποθετεί τα αντικείμενα και ορίζει το μέγεθός τους δίνοντας αναλογικές ή απόλυτες τιμές. Ας δούμε το παρακάτω παράδειγμα του Aabsolutelayout που βρίσκεται στο φάκελο Layouts:

```
<AbsoluteLayout>
    <Label Text="I'm centered on iPhone 4 but no other device"
      AbsoluteLayout.LayoutBounds="115,150,100,100" LineBreakMode="WordWrap"  />
    <Label Text="I'm bottom center on every device."
      AbsoluteLayout.LayoutBounds=".5,1,.5,.1" AbsoluteLayout.LayoutFlags="All"
      LineBreakMode="WordWrap"  />
    <BoxView Color="Olive"  AbsoluteLayout.LayoutBounds="1,.5, 25, 100"
      AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Red" AbsoluteLayout.LayoutBounds="0,.5,25,100"
      AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,100,25"
      AbsoluteLayout.LayoutFlags="PositionProportional" />
</AbsoluteLayout>
```

Η ιδιότητα *AbsoluteLayout.LayoutBounds* ορίζει τα όρια και το μέγεθος του αντικειμένου με την εξής σειρά: X, Y, πλάτος, ύψος. Το LineBreakMode=”WordWrap” σημαίνει ότι όταν το κείμενο του label ξεπεράσει τα όρια άλλαξε γραμμή.
Η ιδιότητα *AbsoluteLayout.LayoutFlags* ορίζει τον τρόπο ερμηνείας των τιμών. Η τιμή All σημαίνει ότι δεχόμαστε τα X,Y,πλάτος και ύψος ως αναλογικά και όχι ως απόλυτες τιμές ενώ η PositionProportional σημαίνει ότι το X,Y ερμηνεύονται αναλογικά ενώ το πλάτος/ύψος με απόλυτες τιμές. Πχ το .5,1 σημαίνει 50% πλάτος/100% ύψος.

Tags: #AbsoluteLayout

Ας δούμε το αποτέλεσμα του παραδείγματος:



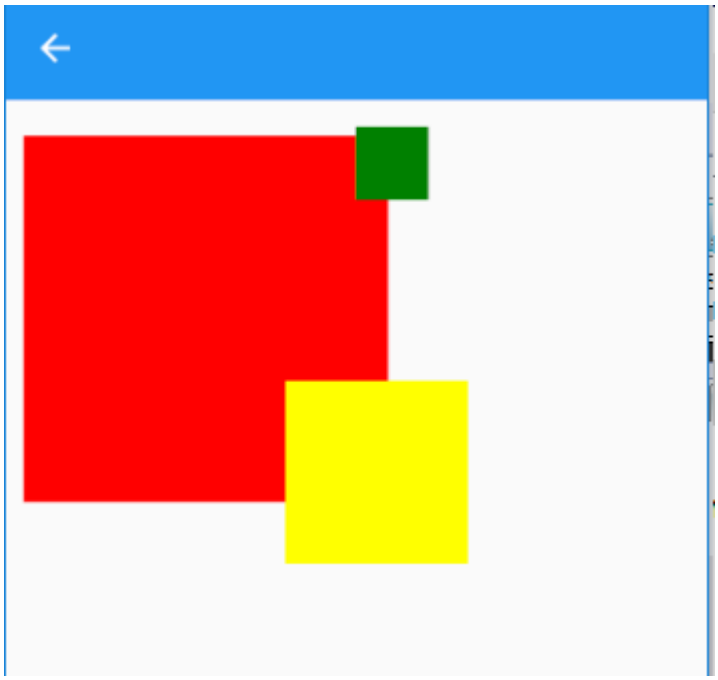
1.11.3. RelativeLayout

Το RelativeLayout χρησιμοποιείται για να ορίσει τη θέση και το μέγεθος των αντικειμένων που περιέχει αναλογικά με τις ιδιότητες του ή τα κοντινά τους στοιχεία. Χρησιμοποιείται για να δημιουργήσουμε προβολές που προσαρμόζονται σε κάθε μέγεθος οθόνης.
Ας ανοίξουμε το αρχείο ARelativeLayout.xaml για να δούμε ένα παράδειγμα:

```
<RelativeLayout>
    <!-- requires x:Name so we can reference in RelativeToView on Green box -->
    <BoxView Color="Red" x:Name="Red"
      WidthRequest="200"
      HeightRequest="200"
       RelativeLayout.XConstraint=
          "{ConstraintExpression Type=Constant,
                      Constant=10}"
       RelativeLayout.YConstraint=
          "{ConstraintExpression Type=Constant,
                      Constant=20}" />
    <BoxView Color="Green"
       RelativeLayout.XConstraint=
          "{ConstraintExpression Type=RelativeToParent,
                      Property=Width,
                      Factor=0.5}"
       RelativeLayout.YConstraint=
          "{ConstraintExpression Type=RelativeToView,
                      Property=Y,
                      ElementName=Red,
                      Constant=-5}" />
    <BoxView Color="Yellow"
      WidthRequest="100"
      HeightRequest="100"
       RelativeLayout.XConstraint=
          "{ConstraintExpression Type=RelativeToParent,
                      Property=Width,
                      Factor=0.4}"
       RelativeLayout.YConstraint=
          "{ConstraintExpression Type=RelativeToParent,
                      Property=Height,
                      Factor=0.3}" />
</RelativeLayout>
```

Tags: #RelativeLayout

Στο πρώτο BoxView θέτουμε WidthRequest και HeightRequest=”200”, δηλαδή δημιουργούμε ένα τετράγωνο 200X200.
Στο κόκκινο κουτί “Red” το RelativeLayout.Xconstraint θέτει τους κανόνες που πρέπει να ακολουθούνται στον άξονα X με μια έκφραση που δηλώνεται με το ConstraintExpression. Ο τύπος είναι μια σταθερά Type=Constant με τιμή Constant=10, δηλαδή μετακίνησέ το 10 μονάδες δεξιά και 20 μονάδες κάτω (δηλώνεται με το Yconstraint). Στο Πράσινο κουτί δηλώνουμε ότι ο περιορισμός στον άξονα X είναι τύπου RelativeToParent δηλαδή θα βρίσκεται στον άξονα X στο 50% του μήκους του RelativeLayout (ακριβώς στη μέση). Επίσης στον άξονα Y, ο περιορισμός είναι τύπου RelativeToView έχοντας απόσταση από την κορυφή του άξονα Y -5 μονάδες από την απόσταση που έχει το στοιχείο Red. Άρα είναι -5 μονάδες πιο ψηλά από το Red.
Στο κίτρινο κουτί δηλώνουμε διαστάσεις 100X100. Στον άξονα X θέτουμε τον περιορισμό να είναι στο 40% του RelativeLayout, ενώ στον άξονα Y να είναι στο 30%.
Ας δούμε στην παρακάτω εικόνα το τελικό αποτέλεσμα.



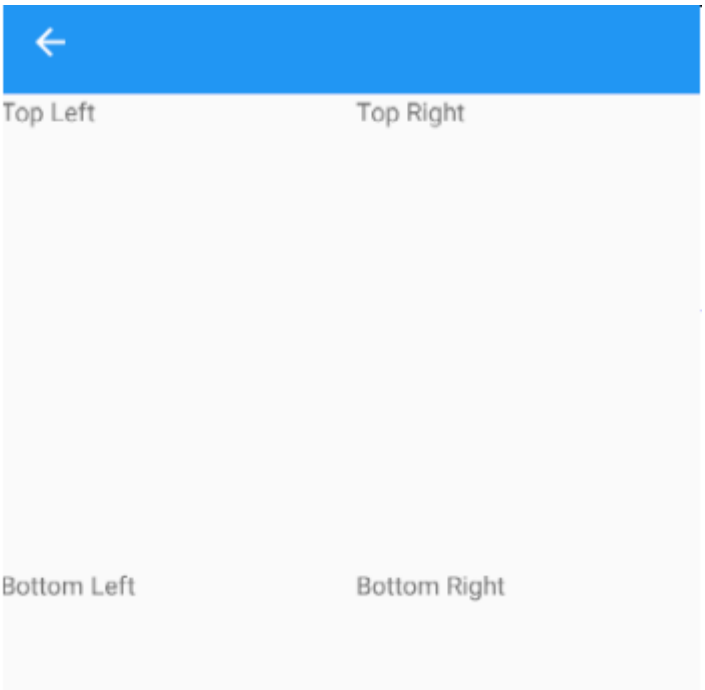
1.11.4. Grid Layout

Με το Grid layout προβάλλουμε τα στοιχεία σε πλέγμα, δηλαδή σε στήλες και γραμμές, όπως ένας πίνακας. Οι διαστάσεις των γραμμών και των στηλών μπορούν να έχουν αναλογικές ή απόλυτες τιμές. Ας μελετήσουμε τον κώδικα του αρχείου AGridLayout.xaml

```
<Grid VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Text="Top Left" Grid.Row="0" Grid.Column="0" />
    <Label Text="Top Right" Grid.Row="0" Grid.Column="1" />
    <Label Text="Bottom Left" Grid.Row="1" Grid.Column="0" />
    <Label Text="Bottom Right" Grid.Row="1" Grid.Column="1" />
</Grid>
```

Tags: #GridLayout

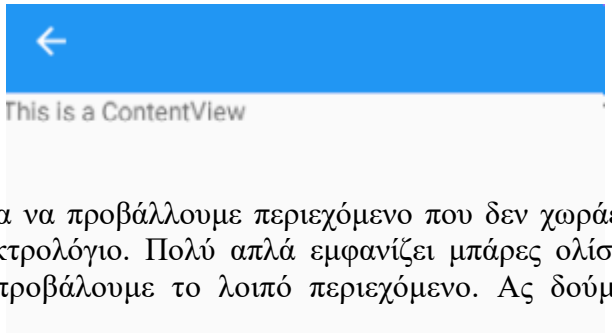
Μέσα στο Grid περιέχονται οι ορισμοί των γραμμών RowDefinitions και των στηλών ColumnDefinition. Παρατηρούμε ότι έχουμε βάλει δύο γραμμές (με το RowDefinition) και δύο στήλες (με το ColumDefinition). Το Height=”*” δηλώνει “πάρε ισάξια με τις άλλες γραμμές που έχουν * τον ίδιο χώρο μέχρι να γεμίσει το layout”. Έπειτα, εισάγουμε τα Label και ορίζουμε τα κελιά στα οποία θα μπουν γράφοντας σε πια γραμμή Grid.Row και σε πια στήλη Grid.Column θα μπει το κάθε ένα. Στην παρακάτω εικόνα φαίνεται το τελικό αποτέλεσμα:



1.11.5. ContentView

Το ContentView είναι το πιο απλό layout καθώς περιλαμβάνει μόνο ένα στοιχείο (πχ μια ετικέτα). Είναι όμως πολύ χρήσιμο καθώς χρησιμοποιείται για τη δημιουργία custom controls του χρήστη. Ένα απλό παράδειγμα υλοποίησης φαίνεται παρακάτω (αρχείο AContentView.xaml) :

```
<ContentView>
    <Label Text="This is a ContentView"></Label>
</ContentView>
```



1.11.6. ScrollView

Το ScrollView χρησιμοποιείται για να προβάλλουμε περιεχόμενο που δεν χωράει στην οθόνη καθώς και να εξασφαλίσουμε χώρο για το πληκτρολόγιο. Πολύ απλά εμφανίζει μπάρες ολίσθησης οριζόντια/κάθετα και επιτρέπει με απλό σύρσιμο να προβάλουμε το λοιπό περιεχόμενο. Ας δούμε τον κώδικα του αρχείου AScrollView.xaml.

```
<ScrollView>
  <StackLayout>
    <BoxView BackgroundColor="Red" HeightRequest="800" WidthRequest="350" />
  </StackLayout>
</ScrollView>
```

Μέσα στο ScrollView έχουμε βάλει ένα άλλο layout τύπου StackLayout, το οποίο περιέχει ένα κόκκινο κουτί μεγέθους 800X350, πολύ μεγαλύτερο από την οθόνη του κινητού. Παρατηρούμε ότι στις άκρες προβάλλονται μπάρες ολίσθησης και με ένα απλό σύρσιμο μπορούμε να προβάλλουμε το κρυφό περιεχόμενο.

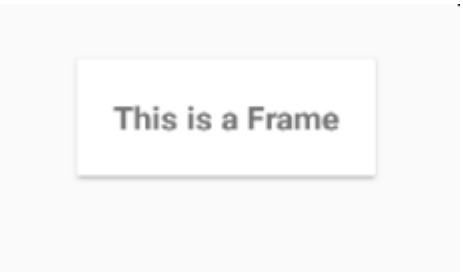
Tags: #ContentView, #ScrollView

1.11.7. Frame

Το Frame είναι ένα πολύ απλό layout που μπορεί να προβάλει ένα αντικείμενο με δυνατότητες πλαισίωσης.

```
<Frame HasShadow="True" OutlineColor="Silver" VerticalOptions="CenterAndExpand"
HorizontalOptions="Center">
        <Label Text="This is a Frame" Font="Bold, Large" HorizontalOptions="Center" />
 </Frame>
```

Όπως φαίνεται στον κώδικα του αρχείου AFrame.xaml μπορούμε να εισάγουμε σκιά με το HasShadow καθώς και χρώμα περιγράμματος με το OutlineColor.



Tags: #Frame