**Xamarin Introduction**  By Kevin Singh

Xamarin.forms is a library that will allow us to create beautiful cross platform mobile applications.

A Xamarin form contains Visual Elements -> Page, Layout, and the View. These are organized in a parent child hierarchy.

Pages are used to hold content. An example of this is the content page.

The Child of a content page is generally a Layout of some sort to organize visuals.

An example of the above is:

We have a Stack layout inside of our content page. This stack layout contains a frame which contains a label or a button. This is basically the structural example of how a Xamarin form is laid out.

We can use one of the following project types for our Xamarin.Forms project. PCL or SAP.

PCL or Portable class Library –

All common app code becomes a dynamic-link library (DLL). That is referenced by individual platform projects. This is what we use since it is easier for a cross platform environment.

SAP or Shared Library –

Loose code fragments that are shared among platform projects.

Usually specified like this –

#define if \_\_IOS\_\_\_

Code to use

#endif

This allows us to define custom codes for specific OS.

Let’s dive into some code fragments

App class which inherits from the Application class. This file contains the instantiation of our content pages.

Example:

namespace Final\_AMD

{

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new MainPage();

}

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

}

}

}

We then have our Content Page:

namespace Final\_AMD

{

public partial class MainPage : ContentPage

{

}

}

The above will be instantiated in our Application class. This will create the main page of our app.

More definitions ->

The **ContentPage** class defines a property called content. Generally, a content is a Layout. An example is the **StackLayout.**

SAP ->

Define code for specific OS

#define if \_\_IOS\_\_\_

Code to use

#endif

PCL ->

Define code for specific OS

Padding = Device.OnPlatform<Thickness>(

new Thickness(0, 20, 0, 0), new Thickness(0), new Thickness(0));

each value for the OnPlatform defines a value for an OS.

IOS is the first one. As you can see in our example.

We can also define a Lambda

Device.OnPlatform (()=>

Padding=new Thickness() , …, … };

Or

Device.OnPlatform(ios:0=>{padding=…});

How this.Padding = Device.OnPlatform<Thickness>(

new Thickness(0, 20, 0, 0), new Thickness(0), new Thickness(0));

ever you are unable to access APIs when using.

The Stack Layout

The contentpage class defines a content property of type view. That you can set to an object. You are only allowed to define one View, per contentpage.

public partial class MainPage : ContentPage

{

This.Content = new stacklayout()…

}

Thus, our stack layout will need to contain all other views.

We can ALSO set the content to hold multiple views if we use a Layout<T>. A list of views.

Layout<views>

Content = layout list

Types of views

Absolute Layout

Grid

Relative Layout

Stack Layout

Simple properties – Content, Orientation, Spacing, ...etc.

To add a child to our stack layout we can use the following

StackLayout.Children.Add(

New Label

{

}

);

Scroll View is auto supported by the Stack View

Adding Scroll to a Stack Layout

Content = new Scrollview

{

Content = StackLayout

};

Button click example –

Button.click += handler option