XAML Essentials

By: Mosh Hamedani

XAML vs Code-behind

- Everything you do with XAML, can be done with code.
- At run-time, XAML parser will process our XAML files and instantiate C# objects.
- We should aim to use XAML for defining the visual appearance of our apps, and use C# for implementing behaviour.

Handling Events

In XAML:

```
<Button Clicked="Handle_Clicked" />
```

In code-behind:

```
private void Handle_Clicked (object source, EventArgs e)
{
}
```

Names

We use **x:Name** attribute to assign an identifier to an element. This will generate a private field that we can access in code-behind or XAML:

```
<Slider x:Name="slider" />
```

Property Element Syntax

We use property element syntax to assign complex objects to attributes.

```
<ContentPage.Padding>
    <OnPlatform x:TypeArguments="Thickness"
       iOS="0, 20, 0, 0"
       Android="0, 40, 0, 0" />
</ContentPage.Padding>
```

Content Property

Some XAML elements (eg **ContentPage** and **StackLayout**) have a *content property*. That means, what we put between their start and end tags will be assigned to their content property. And with this, we don't need to explicitly specify the content property. So, instead of

```
<ContentPage>
    <ContentPage.Content>
        <Label />
    </ContentPage.Content>
</ContentPage>
```

We can simply use:

```
<ContentPage>
    <Label />
</ContentPage>
```

Data Binding

```
<Label Text="{Binding
     Source={x:Reference slider},
     Path=Value,
     StringFormat='Value is {0:F2}'
   }" />
```

If we we have multiple bindings to an object, we can simplify our binding expressions by setting **BindingContext**:

```
<Label BindingContext="{x:Reference slider}"
    Text="{Binding Value}"
    Opacity="{Binding Value}" />
```

When **BindingContext** applied to a container element, it'll be inherited by all its children. This is useful if we have multiple elements in a **StackLayout** (or a **ContentPage**) that reference the same object as their binding source.

XAML Compilation

To compile our XAML files, we add this in **Properties/AssemblyInfo.cs:**

```
[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
```

Dealing with Device Differences

To differentiate between various devices in code-behind, we can use **Device.OnPlatform()** method:

```
Padding = Device.OnPlatform<Thickness>(
   iOS: new Thickness(0, 20, 0, 0)
   Android: new Thickness(0),
   WinPhone: new Thickness(0));


Device.OnPlatform(
   iOS: () => {
       // code to run in iOS only
});
```