Macoratti.net

Xamarin.Forms - Apresentando Estilos : Conceitos - I

Neste artigo vou apresentar o conceito de **estilos** em aplicações Xamarin Forms.

Há algum tempo atrás eu publiquei um artigo no site apresentando o conceito de **Resources** e **ResourceDictionary** em aplicações Xamarin Forms. Vamos aproveitar o conceito de resources e expandir nosso conhecimento para poder criar interfaces mais amigáveis com pouco trabalho.

Ao projetar sua interface de usuário, em algumas situações, você pode ter várias views como Labels, Buttons, etc., do mesmo tipo e, para cada uma delas, você precisará atribuir as mesmas propriedades com os mesmos valores.

Por exemplo, você pode ter dois botões com a mesma largura e altura, ou duas ou mais etiquetas com a mesma largura, altura e configurações de fonte. Em tais situações, ao invés ficar atribuindo as mesmas propriedades muitas vezes, você pode utilizar o recurso dos **estilos ou styles**.

Um **estilo** permite que você atribua um conjunto de propriedades às views do mesmo tipo. Eles devem ser definidos dentro de um **ResourceDictionary**, e devem especificar o tipo para o qual se destinam e um devem possuir um identificador. Assim, aproveitamos o dicionário de recursos do aplicativo para tornar o estilo disponível para todos os controles.

No trecho de código a seguir temos um exemplo de como definir um **estilo** para views Label:

```
<ResourceDictionary>
    <Style x:Key="labelStyle" TargetType="Label">
        <Setter Property="TextColor" Value="Orange" />
        <Setter Property="FontSize" Value="Large" />
    </Style>
</ResourceDictionary
```

Vamos entender o código acima:

Você atribui um identificador usando a expressão **x: Key** e o tipo de destino com **TargetType**, passando o nome do tipo para a view de destino.

Os valores das propriedades são atribuídos usando os elementos **Setter**, cuja propriedade **Property** representa o nome da propriedade de destino e o **Value** representa o valor da propriedade.

Após isso basta atribuir o estilo às views Label da seguinte maneira:

```
<Label Text="Informe o seu nome:" Style="
{StaticResource labelStyle}"/>
```

Dessa forma, um estilo é aplicado atribuindo a propriedade **Style** a uma view com uma expressão que inclui a extensão de marcação **StaticResource** e o identificador de estilo(**labelStyle**) dentro de chaves.

Muito simples não é mesmo ???

Herança de Estilos

Os estilos dão suporte a herança e assim, você pode criar um estilo que deriva de outro estilo. Por exemplo, você pode definir um estilo que segmenta o tipo de abstrato **view** da seguinte maneira:

```
<Style x:Key="viewStyle" TargetType="View">
   <Setter Property="HorizontalOptions" Value="Center" />
   <Setter Property="VerticalOptions" Value="Center" />
</Style>
```

Este estilo pode ser aplicado a qualquer view, independentemente do seu tipo concreto. Então você pode criar um estilo mais especializado usando a propriedade **BasedOn** da seguinte maneira:

```
<Style x:Key="labelStyle" TargetType="Label"   BasedOn="{StaticResource viewStyle}">
    <Setter Property="TextColor" Value="Orange" />
</Style>
```

O segundo estilo, **labelStyle**, se destina a ser usado com views Label, mas também herda as configurações de propriedade do estilo pai.

De forma resumida, o estilo **labelStyle** irá atribuir as propriedades **HorizontalOptions, VerticalOptions e TextColor** às views Label.

Aplicando estilos implícitos

A propriedade **Style** das views permite a atribuição de um estilo definido dentro dos recursos. Isso permite que você atribua seletivamente o estilo apenas a determinadas views de um determinado tipo.

No entanto, se você quizer que o mesmo estilo seja aplicado a todas as views do mesmo tipo na interface do usuário, atribuir a propriedade **style** para cada view manualmente pode ser um trabalho entediante.

Neste caso, você pode aproveitar o recurso chamado **estilo implícito**. Este recurso permite que você atribua automaticamente um estilo a todos as views do tipo especificado com a propriedade **TargetType** sem a necessidade de configurar a propriedade **Style**.

Para conseguir fazer isso, você simplesmente evita atribuir um identificador usando **x:key**, como na exemplo a seguir:

```
<Style TargetType="Label">
   <Setter Property="HorizontalOptions" Value="Center" />
   <Setter Property="VerticalOptions" Value="Center" />
   <Setter Property="TextColor" Value="Orange" />
</Style>
```

Observe que definimos um estilo especificando o **TargetType** e definindo **Property** e **Value** para as propriedades mas não atribuimos um identificador usando **x:Key**.

Os Estilos definidos sem um identificador serão aplicados automaticamente a todas as views **Label** na interface do usuário *(de acordo com o escopo do resource dictionary)* e você não precisará atribuir a propriedade **Style** nas definições de cada Label.

Assim usando estilos com o recurso **StaticResources** temos uma ótima maneira de reduzir os valores duplicados,

mas o que precisamos é a capacidade de alterar o dicionário de recursos em tempo de execução e ter as as atualizações de recursos refletidas quando referenciadas.

Para isso podemos usar **DynamicResource** nas chaves de dicionário associadas a valores que podem ser alterados durante o tempo de execução. Além disso, ao contrário dos recursos estáticos, os recursos dinâmicos não geram uma exceção em tempo de execução se o recurso for inválido e simplesmente usará o valor de propriedade padrão.

Na **próxima parte do artigo** vamos aplicar os conceitos apresentandos mostrando na prática como usar estilos.

(Disse Jesus) O meu mandamento é este: Que vos ameis uns aos outros, assim como eu vos amei.
João 15:12

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**

- Jose C Macoratti (@macorati) | Twitter
- **An Introduction to Xamarin.Forms - Xamarin**
- **https://www.visualstudio.com/pt-br/features/xamarin-vs.aspx**
- **https://xamarin.com/starter**
- **Xamarim Studio - Desenvolvimento Multiplataforma com C# (Android, iOS e Windows)**
- **Xamarim - Criando Apps com o Visual Studio e C#** **(vídeo aula)**
- **https://developer.xamarin.com/guides/xamarin-forms/xaml/xaml-basics/data_binding_basics/**
- **https://developer.xamarin.com/guides/xamarin-forms/user-interface/styles/**

José Carlos Macoratti