12

ListView - Binding

Transcrição

Com o `ListView` mostrando a listagem de veículos, ficam faltando os preços, portanto precisaremos modificar a aplicação para incluí-los. Para deixar o código mais organizado e "correto", criaremos uma classe que os armazenará, tendo duas propriedades:

```
namespace TestDrive
{
    public class Veiculo
    {
        public string Nome { get; set; }
        public decimal Preco { get; set; }
    }

    public partial class MainPage : ContentPage
    {
        public List<Veiculo> Veiculos { get; set; }

        public MainPage()
        {
            InitializeComponent();

            this.Veiculos = new List<Veiculo>
            {
                new Veiculo { Nome = "Azera V6", Preco = 60000 },
                new Veiculo { Nome = "Fiesta 2.0", Preco = 50000 },
                new Veiculo { Nome = "HB20 S", Preco = 40000}
            };

            listViewVeiculos.ItemsSource = new string[]
            {
                "Azera V6",
                "Fiesta 2.0",
                "HB20 S"
            };
        }
    }
}
```

Agora que temos a lista criada e populada, alteraremos o `ItemsSource` para apontar a nova origem de dados da `listViewVeiculos` para a lista de veículos.

```
namespace TestDrive
{
    public class Veiculo
    {
        public string Nome { get; set; }
        public decimal Preco { get; set; }
    }
```

```
    public partial class MainPage : ContentPage
    {
        public List<Veiculo> Veiculos { get; set; }

        public MainPage()
        {
            //...

            listViewVeiculos.ItemsSource = this.Veiculos;
        }
    }
}
```

Com a aplicação sendo rodada, verifica-se que em vez de trazer os nomes dos veículos, o que se exibe na tela agora são três linhas com "TestDrive.Veiculo" simplesmente, ou seja, com o nome do objeto, mas não o nome de cada veículo.

O `ListView` ainda é muito simples, e não trata a propriedade de cada objeto do tipo `Veiculo`. Então, o objeto é convertido em uma `string`. Quando tínhamos um *array* de `strings`, cada item destes é uma `string` própria, fácil de ser exibida. Como temos como `ItemsSource` uma lista de veículos, exibe-se o veículo, e sua propriedade, porém esta não é utilizada.

Abrindo-se o arquivo `MainPage.xaml`, modificaremos o `ListView` para que se acesse a propriedade `Nome` de cada veículo. Definiremos algumas propriedades internas à tag `<ListView>` para a correta exibição do nome do veículo. Para isto, utilizaremos a propriedade `ItemTemplate` para mostrar cada elemento.

`DataTemplate` irá conter os controles do Xamarin Forms que vão exibir o dado na tela, função do `ViewCell`, uma "célula de exibição", traduzido para o português. Dentro dele, teremos que definir a propriedade `ViewCell.View`, o qual, por sua vez, possui o `Label`. Ali, definiremos o texto a ser exibido.

Já sabemos que o `ItemsSource` da `ListView`, sua fonte de dados, é uma lista do tipo `Veiculos`. Ou seja, cada elemento ou linha deste `ListView` exibirá um veículo, cuja propriedade `Nome` passaremos ao `Label`.

Faremos isto utilizando uma técnica chamada `Binding`, que em inglês significa "colagem", ou "amarração". Ele fará com que o `Label` "se agarre" a uma determinada propriedade do objeto que está recebendo, neste caso, o `Veiculo`.

```
<ListView x:Name="listViewVeiculos">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <ViewCell.View>
                    <Label Text="{Binding Nome}"></Label>
                </ViewCell.View>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Agora rodaremos a aplicação para ver seu comportamento, com a correta exibição do nome dos veículos inicializados na listagem. Ainda falta acrescentarmos o valor de cada veículo, o que faremos em breve.

Modificaremos a maneira como informamos ao `ListView` os veículos exibidos na tela, então em vez de utilizarmos o `ItemsSource`, usaremos outra maneira, tal qual o `Binding`, o qual usaremos no `ItemsSource`, pegando uma coleção de

dados a ser exibido no `ListView`. Em `MainPage.xaml`, então, teremos:

```
<ListView x:Name="listViewVeiculos" ItemsSource="{Binding Veiculos}">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <ViewCell.View>
                    <Label Text="{Binding Nome}"></Label>
                </ViewCell.View>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Feito isto, removeremos o `ItemsSource` de `MainPage.xaml.cs` (a linha `listViewVeiculos.ItemsSource = this.Veiculos;`), e rodaremos a app. Não aparece nada! Isto ocorre porque o `Binding` de `MainPage.xaml` pega uma propriedade denominada `Veiculos` trazendo-a para o `ListView` do XAML, porém ele ainda não sabe de onde trazer estes dados do veículo.

Precisamos dizer à `MainPage.xaml` o contexto de `Binding` que deverá ser utilizado para passar as propriedades ao `Binding` do XAML, em que acrescentaremos:

```
this.BindingContext = this;
```

Em que o `BindingContext` é a própria classe `MainPage.xaml.cs`, o próprio *code behind*, que servirá como origem para a amarração que queremos utilizar no XAML. Rodaremos a aplicação novamente, e o resultado obtido foi a correta exibição do nome dos veículos.

Tivemos bastante informação neste vídeo, não é verdade? Espero que vocês estejam gostando do Xamarin Forms, muito obrigado e até a próxima!