05

EditarPerfilCommand

Transcrição

A `MasterView`, no momento, é uma `TabbedPage`, uma página com abas. Na aba "Usuário", temos um botão "Perfil", que por enquanto não possui nenhuma função. Porém gostaríamos que ele nos levasse à aba "Editar" (de edição dos dados do usuário) programaticamente, via código C#, sem que haja necessidade de se clicar diretamente nesta aba.

Como este botão não possui nenhum evento programado, ou comando `Binding` associado, teremos que implementar isto agora. Conforme fizemos anteriormente, criando-se comandos para cada um dos botões da aplicação, precisamos localizar a `ViewModel` desta *view*. Então, para `MasterView`, precisamos encontrar `MasterViewModel`, que se localiza em `MasterViewModel.cs`, a qual abriremos para declarar uma nova propriedade que vai expor à *view* o comando a ser executado assim que o usuário clicar no botão "Perfil".

Utilizaremos o atalho "`propg`" que cria o corpo de uma propriedade a partir de `get` público e `set` privado, por meio do qual não conseguimos setar a propriedade `EditarPerfilCommand` fora da classe, apenas internamente. Definiremos o código a ser executado quando este comando for acionado pela *view*, no construtor da classe.

O `EditarPerfilCommand` será um novo `Command`, dentro do qual passaremos uma `Action` a ser executada quando o usuário apertar o botão. Tal `Action` pode ser um método anônimo com uma expressão lambda, de que não recebemos nenhum parâmetro, e cujo corpo será vazio.

```
private readonly Usuario usuario;

public ICommand EditarPerfilCommand { get; private set; }

public MasterViewModel(Usuario usuario)
{
    this.usuario = usuario;

    EditarPerfilCommand = new Command(() =>
    {

    });
}
```

Assim, clicaremos em `MasterViewModel.cs` no menu lateral do Visual Studio para "amarrar" (por meio do `Binding`) este comando à ação de clicar no botão. Definiremos a propriedade `Button` como sendo `Command`:

```
<Button Text="Perfil" Command="{Binding EditarPerfilCommand}"></Button>
```

Com isto temos o necessário para executarmos o código. Rodando a aplicação, vamos ver o que acontece quando clicamos no botão "Perfil" da página de mesmo nome. A app nos leva ao `EditarPerfilCommand`, que ainda não possui nenhum código atrelado a si, porém, como colocamos um *breakpoint* ali, o programa parou naquele momento, confirmando que o comando está sendo executado corretamente.

Agora precisaremos programar o que deve ser feito quando este botão for clicado para que quando clicamos em "Perfil" sejamos redirecionados à aba "Editar". Antes, decidiremos de que maneira informaremos ao programa sobre o clique

no botão. É necessário enviar uma mensagem, chamada `EditarPerfil`, que será o tipo da mensagem a ser enviada via sistema de mensagens do Xamarin Forms, o `MessagingCenter`.

Como estamos trabalhando com o usuário, queremos que a mensagem tenha como anexo um objeto de tipo `Usuario`.

Dentro dela, passaremos uma instância deste objeto para chegarmos ao corpo da `Action` desta mensagem (`usuario`). O método `Send` também exige o nome da mensagem, que será seu tipo e também será responsável por capturar esta mensagem (`EditarPerfil`).

```
private readonly Usuario usuario;

public ICommand EditarPerfilCommand { get; private set; }

public MasterViewModel(Usuario usuario)
{
    this.usuario = usuario;

    EditarPerfilCommand = new Command(() =>
    {
        MessagingCenter.Send<Usuario>(usuario, "EditarPerfil");
    });
}
```

Para capturar esta mensagem após seu envio, precisamos de uma assinatura que permita a troca da aba, o que ocorrerá em um lugar específico do código. Trocaremos a aba da *view* exatamente no *code behind*, justamente onde temos a capacidade de acessar o objeto `TabbedPage`, trocando a aba atual. Portanto, em `MasterView.xaml.cs`, faremos a assinatura daquela mensagem, capturando-a, usando o método `override`:

```
public partial class MasterView : TabbedPage
{
  public MasterView(Usuario usuario)
  {
      InitializeComponent();
      this.BindingContext = new MasterViewModel(usuario);
  }

  protected override void OnAppearing()
  {
      base.OnAppearing();

      MessagingCenter.Subscribe<Usuario>(this, "EditarPerfil",
        (usuario) =>
        {
          this.CurrentPage = this.Children[1];
        });
  }

  protected override void OnDisappearing()
  {
    base.OnDisappearing();

    MessagingCenter.Unsubscribe<Usuario>(this, "EditarPerfil");
  }
}
```

No código acima, temos como último parâmetro de `Subscribe` um *callback* que será executado quando a mensagem for interceptada, dentro do qual passaremos outro lambda que será uma `Action`, desta vez com parâmetros, que recebe um `usuario` sendo enviado juntamente com a mensagem. Dentro dele, tem-se o código que roda quando a mensagem é interceptada, quando a aba é trocada.

Como também precisamos cancelar a assinatura desta mensagem quando a mensagem sumir da tela, colocamos outro `override`, com `OnDisappearing`, para o qual copiaremos e colaremos o método utilizado anteriormente, removendo o *callback* e utilizando `Unsubscribe` em vez de `Subscribe`.

Para a troca de aba, observamos que o objeto da classe `MasterView` é um `TabbedPage`, que possui, além das páginas-filhas (uma coleção de páginas), uma propriedade que define a página atual (`CurrentPage`). Sendo a aba "Usuário" a primeira página, a aba "Editar" é a segunda. Estas páginas possuem índice base `0`, então "Usuário" seria a página-filha de índice `0` e, "Editar", a página-filha de índice `1`, a segunda página da `TabbedPage` (como indica `this.Children[1]` no código).

Rodaremos a app e logaremos. Depois, clicando em "Perfil" na página de perfil do usuário, localizada na primeira aba, somos redirecionados à aba "Editar", conforme gostaríamos.