06

Button, Command e ICommand

Transcrição

[00:00] Bom, então a gente acabou de ver como utilizar mensagens para fazer comunicação entre os componentes. Que componentes são esses? É view e viewmodel. A gente fez isso para a primeira página, que é a página de listagem, a gente conseguiu remover com sucesso a chamada para um evento do code behind, que é o evento ItemTappet.

[00:20] A gente trocou isso por uma propriedade que é o selected item e a gente fez o binding para essa propriedade lá no nosso viewmodel e agora a gente vai ver como fazer isso além, a gente vai começar a utilizar nas outras páginas também. Então aqui na segunda página, que é a página de detalhe...

[00:41] A gente tem esse botão aqui “próximo”, esse botão quando clicado, vai navegar, vai levar o usuário para a próxima página, que é a página de agendamento. Esse botão “próximo”, ele também está amarrado a um evento no code behin.

[00:59] Então a gente vai desacoplar isso também, utilizando as mensagens para a gente poder fazer a notificação para a view, entre a view e a viewmodel, e aí, sim, a view vai fazer a navegação através da mensagem. Então a gente vai parar aqui a nossa aplicação, a gente vai entrar em detalhe view.

[01:26] E olha só, em detalhe view, eu tenho botão, que é o botão “próximo”, aqui em baixo e eu vou trocar esse evento clicked, por um outro conceito, que não é mais um evento, a gente vai trocar o evento por um comando. E como é comando em leis, então é command.

[01:49] Então, eu coloco aqui um command, e aí, eu vou fazer a programação dessa comando. Para a gente programar o comando, a gente pode simplesmente utilizar o nosso conhecido binding e esse binding vai acionar alguma coisa lá no nosso viewmodel.

[02:10] A gente vai chamar um novo tipo de objeto chamado comando ou command, esse comando vai executar uma ação. Então, a primeira a fazer aqui, é remover o evento o clicked. Então, eu vou começar a trabalhar com comandos agora e aqui no command, eu coloco binding.

[02:33] E aqui, eu vou dar um nome para o comando e essa referência, esse objeto de comando vai existir no nosso viewmodel. Então, eu vou chamar ele de ProximoCommand. Então, eu vou pegar esse ProximoCommand e vou jogar, vou criar uma nova propriedade lá no nosso viewmodel.

[02:58] Então a viewmodel, que é o detalhe viewmodel, aqui eu vou colocar uma referência, aqui em baixo, public, eu vou colocar um ICommand e vou colocar o nome da nossa referência que é o próximo command, que vai ser uma propriedade. Então, eu coloco aqui: get; set;. Ficando: public ICommand ProximoCommand {get; set; }.

[03:26] Então, eu defini uma nova propriedade chamada próximo command. Agora, como que a gente faz para utilizar um comando com o Xamarin Forms? A gente tem que pegar esse próximo command e instanciar ele. E como que eu faço isso? Como que eu instancio um comando dentro de um viewmodel?

[03:47] Bom, eu pego esse nome, dessa propriedade e vou lá para o nosso construtor do viewmodel, deixa eu procurar aqui o construtor, achei. Eu vou programar esse construtor para inicializar o nosso comando. Então, agora eu vou inicializar o próximo comando.

[04:27] Então, eu vou colocar aqui new Command e vou passar aqui dentro, qual vai ser o mando que ele vai executar e para simplificar aqui a nossa vida, vou colocar aqui dentro um método anônimo, vou colocar uma expressão lambda e aqui dentro vou executar o comando. Agora, que comando é esse?

[04:34] Quando ele clicar nesse botão, no próximo, eu quero que ele lance uma mensagem, para ele notificar a view de detalhe, que o botão próximo foi acionado. Então, eu vou fazer isso utilizando o nosso centro de mensagens do Xamarin Forms, que é o MessengerCenter e vou enviar uma mensagem.

[05:00] Como que eu envio uma mensagem? Com o método send. Então, aqui dentro, eu vou passar também a referência, a quem está enviando, que é this, que é a própria classe e aqui o nome da mensagem. Então, eu vou chamar essa mensagem de próximo.

[05:26] Então, eu vou enviar uma mensagem chamada próximo e em algum lugar a aplicação vai ter que tratar essa mensagem. Agora a gente vai lá no nosso detalhe view, lá no code behind e vai ter que capturar, vai ter que tratar essa mensagem que foi enviada pela viewmodel.

[05:45] Então, lembrando que a nossa view não sabe quem enviou a mensagem, os componentes não se conhecem, eles são sabem que está enviando a mensagem para ela e quem está recebendo também. Como a gente viu antes, a gente programa o nosso método de assinar a mensagem, preferencialmente no nosso evento OnAppearing.

[06:12] Então, eu coloco aqui protected override OnAppearing, que é o momento que a página está aparecendo e que ela vai assinar essa mensagem que está sendo recebida. Então, eu coloco aqui MessagingCenter.Subscribe, para ele poder capturar a mensagem.

[06:39] Eu coloco aqui a referência de quem está assinando, que é a própria classe do code behind e aqui o nome da mensagem. Então aqui, próximo e agora, eu vou colocar também o callback. Então aqui dentro, eu coloco um método anônimo que vai ser acionado quando a mensagem for recebida.

[07:05] Bom, então aqui eu fiz errado, eu coloquei subscribe, mas eu não informei qual é o tipo, que é o veículo, que é a mensagem que está sendo enviada. Então, quando eu clico no próximo, eu também tenho que passar o veículo que está indo para a próxima página. Ficando: MessagingCenter.Subscribe(this, “Proximo”, (m) ->.

[07:21] Agora, eu vou voltar lá para o detalhe viewmodel e vou passar aqui no send, eu vou colocar o veículo que foi selecionado, que é o veículo. Ficando: MessagingCenter.Send(veiculo, “próximo”). Então, voltando aqui, eu tenho o OnAppearing e aqui ele vai...

[07:41] Eu vou ter que colocar o código que vai navegar para a próxima, que é página de agendamento. Então, eu vou simplesmente pegar esse código que foi criado para o efeito clicked, do botão e vou mover ele aqui para baixo. Então, olha só, estou fazendo a navegação usando o PushAsync...

[08:04] Instanciando um novo agendamento view, uma nova página de agendamento, eu vou passar aqui dentro a mensagem que está sendo recebida, que é o veículo que está sendo escolhido durante o processo do fluxo da aplicação. Agora eu posso remover o evento buttomProximo_Clicked.

[08:28] Com isso, eu removo essa dependência do code behind com relação a nossa view. Agora, ainda falta fazer o unsubscribe, fazer esse cancelamento da assinatura quando a página desaparecer da tela, a gente faz isso com o pretected override void OnDisappearing.

[08:54] A gente vai fazer aqui MessagingCenter.Unsibscribe e eu passo aqui o tipo do argumento da mensagem, que é o veículo, eu passo a referência this, que é quem está cancelando a assinatura e aqui eu passo o nome da mensagem, que é próximo. Ficando: MessagingCenter.Unsubscrive(this, “Proximo”);.

[09:17] Agora que a gente fez isso, você pode se perguntas: mas como é que as mensagens são trocadas, são enviadas e são recebidas, como é que uma mensagem não é confundida com outra mensagem? Bom, isso é fácil, porque cada mensagem tem um nome, tem um tipo.

[09:36] Então essa mensagem aqui, nesse caso, se chama próximo, a outra mensagem que a gente criou antes chama veículo selecionado. Então, quando a gente assina uma mensagem chamada próximo, então esse método aqui dentro, esse código aqui dentro, ele só vai ser executado, quando uma mensagem do tipo próximo for enviada.

[10:00] Então, nunca uma mensagem do tipo veículo selecionado vai acionar esse código, porque justamente ele tem um nome diferente. Então, com os nomes diferentes, a gente consegue ter ações diferentes, então a gente evita essa confusão das mensagens, por causa do nome de cada uma delas.

[10:21] Feito isso, a gente já tem aqui o código pronto para o nosso botão, então a gente vai executar a aplicação e ver se ele está funcionando corretamente. Então, agora está rodando o programa, vou selecionar aqui Fiesta 2.0. Vou clicar no próximo, muito bem.

[10:45] Então, a gente conseguiu navegar da primeira para a segunda, da segunda para a terceira página utilizando o MVVM, utilizando mensagem e utilizando agora também o comando que faz parte do Xamarin Forms.