03

Mensageria, MessagingCenter, OnAppearing e On Disappearing, Subscribe e Unsubscribe e Send

Transcrição

[00:00] Bom, como a gente acabou de ver, eu não posso simplesmente utilizar aqui uma referência para o objeto navigation dentro do nosso viewmodel, por quê? Porque o viewmodel não é uma página e ele não tem esse objeto exposto, que ele possa ser utilizado para a gente navegar.

[00:21] Então, o que a gente tem que fazer? A gente tem que de alguma forma avisar a nossa listagem view que a gente quer navegar para a próxima página. Só que, olha só, se eu utilizar uma referência aqui para a view, a parte da listagem viewmodel.

[00:39] Então, eu estou quebrando o conceito do MVVM, porque eu vou usar uma referência para uma view, dentro de um viewmodel, então eu não quero isso, eu não quero acoplar mais o código, eu não quero gerar mais dependências dentro da nossa aplicação. Então, o que eu posso fazer aqui é utilizar um conceito chamado mensageria.

[01:03] Então, eu posso, por exemplo, emitir uma mensagem nova e essa mensagem vai ser recebida por outro lugar, outro componente, que no caso vai ser a view. Então, a gente joga uma mensagem no ar, essa mensagem vai ficar por aí, pela aplicação e vai ser recebida por quem interessar essa mensagem.

[01:25] Então, o que a gente vai fazer aqui? É lançar uma mensagem a partir da nossa listagem viewmodel e aí, quem tiver o interesse nessa mensagem, que no caso é a nossa view, vai pegar essa mensagem, vai tratar e vai fazer a navegação. Então, o que eu vou criar aqui é uma referência para um objeto global que existe no Xamarin Forms...

[01:48] Que é o centro de mensagem, é o centro de mensageria e ele é chamado de MessagingCenter e eu vou fazer o quê? Eu vou enviar uma nova mensagem. Então, eu coloco aqui send e coloco aqui também uma instância da mensagem que eu vou criar. Então, olha só, que mensagem eu vou enviar?

[02:19] Essa mensagem se chama veículo selecionado, por quê? Porque olha, eu estou enviando uma mensagem para a view, dizendo o seguinte: “Olha, eu estou selecionando o veículo, então tem um veículo selecionado”. Então, eu vou chamar essa mensagem de VeiculoSelecionado.

[02:37] Só que eu tenho que passar também nessa mensagem quais são os argumentos, o que eu estou mandando nessa mensagem, eu não posso simplesmente só falar que eu selecionei algo, eu tenho que falar também o que foi selecionado.

[02:51] Então, eu vou (passar) aqui o veículo, que é o veículo que está no nosso viewmodel, aliás, é o veículo selecionado. Então, eu vou copiar aqui essa propriedade, vou passar aqui em baixo, ficando: MessagingCenter.Send(veiculoSelecionado, “VeiculoSelecionado”).

[03:05] Então, agora que a mensagem vai ser enviada, ela também precisa ser recebida, então, quem vai receber essa mensagem? Quem receber a mensagem é aquele que tem a capacidade de trocar de página, aliás, é aquele componente que pode fazer o usuário navegar para uma segunda página, que é a nossa view.

[03:28] Então, eu vou entrar aqui em listagem view e vou ter que assinar, vou ter que subscrever essa mensagem, porque esse componente, essa view, ela vai receber a mensagem e vai fazer o tratamento para navegar para a próxima página. Então, a gente vai fazer isso agora.

[03:49] Então, para subscrever essa mensagem, para assinar essa mensagem, eu poderia fazer esse código em qualquer lugar aqui da nossa classe, poderia, por exemplo, colocar aqui no construtor. Poderia colocar: MessengerCenter.Subscribe.

[04:07] E passando aqui o tipo do argumento que vai ser passado, que no caso é veículo, poderia passar aqui this... e aqui, continuar, colocar o código para fazer a nossa assinatura, ficando: MessengerCenter.Subscribe(this.…), mas invés disso, eu vou colocar o nosso código que vai assinar a nossa mensagem num lugar mais apropriado.

[04:30] Que é... no momento em que a página está aparecendo na tela. Então para isso, eu vou entrar aqui na classe e vou começar a codificar um método chamado OnAppearing, que significa ao aparecer. Então, eu coloco aqui protected override e aqui eu procuro o OnAppearing.

[04:55] E aqui dentro, sim, é que eu vou começar a colocar o código que vai subscrever, que vai assinar essa mensagem de que algum veículo foi selecionado, tá bom? Então eu coloco aqui MessengerCenter.Subscribe e aqui eu tenho que colocar o tipo do argumento que está sendo enviado para dentro dessa mensagem.

[05:16] Então qual o conteúdo dessa mensagem? É o veículo que foi selecionado, então o tipo é veículo e aqui eu passo quem é a instância que está subscrevendo, que está assinando essa mensagem e a instância, a gente coloca aqui a referência this, porque é o próprio code behind que está assinando essa mensagem.

[05:40] E aí, aqui eu coloco qual é o nome da nossa mensagem, a gente coloca a mensagem lá como VeiculoSelecionado, cuidando para não trocar maiúsculo por minúsculo, porque isso vai afetar a execução e a sua mensagem pode não ser capturada, porque você não utilizou corretamente o maiúsculo e minúsculo, ficando: MessengerCenter.Subscribe(this, “VeiculoSelecionado”).

[06:05] Agora, o próximo argumento aqui do nosso... do callback, é uma função anônima que vai receber a mensagem e tratar e fazer a execução, para navegar para a próxima página. Então aqui, eu vou passar uma variável msg, um parâmetro msg e aqui dentro, eu vou fazer a execução para navegar para a próxima página.

[06:33] Então, note que isso aqui é uma expressão lambda, que vai executar quando a mensagem for recebida, quando ela for capturada pela view. Então, vamos lá. Agora, aqui, eu vou fazer msg., e aqui, olha só que interessante, eu tenho acesso a todas as propriedades que estão lá na nossa mensagem, que estão lá no nosso veículo.

[06:58] Então, com isso, eu consigo pegar esses dados e mandar para a próxima página. Então eu vou colocar aqui um breakpoint, só para a gente poder rodar esse código do jeito que está e poder testar. Só que, olha só, nessa mudança, ele está (setando) o valor do veículo selecionado para nulo.

[07:31] Mas será que a gente quer notificar a view de que o veículo selecionado é nulo? Eu acho que não, não faz sentido. Então, eu vou parar a aplicação aqui e vou colocar uma condição. Então, se o valor for diferente de nulo, aí sim eu quero notificar. Então, vou tirar o breakpoint daqui e vou colocar aqui em baixo.

[07:56] Vamos rodar de novo. Rodando a aplicação agora, vamos ver o que acontece. Legal, agora ele não acionou mais o veículo selecionado, então agora eu vou selecionar, vou tocar aqui no Fiesta 2.0. Agora que eu toquei, ele caiu aqui na linha que ele vai enviar a mensagem pelo MessengerCenter.

[08:23] Então, o veículo selecionado aqui é o nosso Fiesta 2.0. Agora, eu vou rodar com o F5 e ele parou. Olha só, ele foi lá no código do code behind, do nosso view, ele parou aqui dentro do OnAppearing, então ele parou aqui dentro dessa expressão lambda, dessa função aqui...

[08:48] Que é onde ele está capturando a mensagem que a gente enviou em outro componente. Então, olha só que interessante, nesse momento, a viewmodel manda uma mensagem, mas não sabe quem é que vai receber essa mensagem, não sabe nem se alguém vai receber essa mensagem.

[09:06] E quem recebe essa mensagem é o view e a view, o code behind também não sabe quem enviou a mensagem. Então, a gente consegue utilizar o MessengerCenter para trocar mensagens entre componentes que não se conhecem. Então, isso é uma maneira de a gente reduzir o acoplamento.

[09:25] De a gente melhorar a arquitetura da nossa aplicação, para reduzir as dependências entre os componentes. Então, agora, dentro dessa captura dessa mensagem, a gente tem esse parâmetro msg, vamos ver o que que ele contém agora. Então, olha só, lá dentro do msg a gente tem um veículo e que veículo é esse?

[09:46] É o mesmo veículo que foi enviado lá pelo nosso viewmodel, que é o Fiesta 2.0. Agora, o que falta fazer, é programar a nossa classe, o nosso code behind para ele... da mesma forma como ele assinou, quando ele subscreveu essa mensagem, ele também tem que "desassinar", fazer o inverso da subscrição...

[10:16] Que é para ele fazer o cancelamento desse registro, dessa assinatura. Então, a gente vai fazer isso com o método chamado OnAppearing. Então a gente programa aqui protected override void OnDisappearing() e aqui na linha de baixo, a gente vai fazer o unsubscribe.

[10:36] Então, a gente colocar aqui: MessengerCenter.Unsubscribe, coloca veículo, que é o tipo da mensagem e a gente coloca aqui o this e o nome da mensagem que é veiculoSelecionado, ficando: MessengerCenter.Unsubscribe(this, "VeiculoSelecionado").

[11:02] Bom, feito isso, ainda falta a gente fazer o usuário navegar para outra página, então falta colocar ainda a chamada para o navigation, essa linha aqui que a gente já tinha antes, feito através do evento ItemTappet, que a gente vai remover agora. A gente vai ter que pegar esse código e colocar aqui dentro da nossa essa pressão lambida, desse método anônimo.

[11:30] A gente vai fazer a navegação. Agora eu vou remover aqui o evento ItemTappet e vou passar no OnAppearing, aqui no subscribe do veículo selecionado, eu vou chamar o detalhe view, que é a segunda página do nosso fluxo, eu vou chamar essa página, passando como argumento a mensagem que está chegando aqui nessa assinatura.

[12:00] E essa mensagem, esse parâmetro msg. Então, eu coloco msg aqui dentro e agora a gente vai rodar a aplicação e ver como ela se comporta. Então, agora está rodando a aplicação, vou selecionar aqui HB20S. Ok, toquei aqui, ele vai enviar uma mensagem.

[12:26] Enviou e olha só, ele foi para a segunda página, a página de detalhe. Selecionando aqui em cima HB20S. Então, com isso a gente conseguiu fazer um ciclo, né? Para pelo menos a primeira página, utilizando o MVVM e utilizando o mensageria, que é suportado pelo Xamarin Forms, que facilita...

[12:50] Com que a gente desacople o nosso código, que a gente consiga trocar mensagens, entre componentes que não sabem quem vai enviar e quem vai receber. Então, com isso a gente simplificou o nosso código ou melhor, a gente conseguiu desacoplar...

[13:08] Conseguiu eliminar dependências de código, que a gente tinha quando a gente utilizava eventos do list view.