13

Revisão

Transcrição

[00:00] Eu sou Marcelo Oliveira, estamos chegando ao final do nosso curso do Xamarin com Visual Studio, parte 02 e nessa parte do curso, a gente atacou dois problemas de arquitetura da aplicação. Esses problemas são: o auto acoplamento e o isolamento da aplicação.

[00:20] E primeiro, a gente viu como desenvolver uma aplicação, utilizando o MVVM. Então, a gente pegou a aplicação já funcionando e a gente desacoplou a nossa view do nosso modelo, utilizando aqui essa classe, essa camada intermediária, que é a viewmodel, que é aonde fica a lógica de apresentação.

[00:43] Com isso, a gente diminuiu a dependência da view, com relação ao modelo, à lógica e aos dados do negócio. Então, a gente introduziu esse componente intermediário, que é a viewmodel. E lá no código, o que a gente fez para isso? Primeiro, a gente pegou para cada uma das views...

[01:03] A gente definiu o contexto de binding. Então, o contexto de binding, a gente apontou lá para a viewmodel, que é a nossa classe de viewmodel correspondente a cada uma das views. Então a gente definiu o binding context como sendo o viewmodel.

[01:22] E, obviamente, a gente também teve que criar uma classe de viewmodel para cada uma das views. Então, por exemplo, para a listagem view, a gente criou a listagem viewmodel. A listagem viewmodel tem a lógica de apresentação. Então, entre outras coisas...

[01:37] Ela permite, ela fornece dados lá para a view e também trata os dados, trata as ações do usuário nessa página, nessa view. E aí, nessas propriedades da viewmodel, a gente... cada vez que essa propriedade é modificada, a gente tem que chamar esse método OnPropertyChanged.

[01:58] Então, OnPropertyChanged, é um método que vai estar lá na classe base para cada uma das viewmodels. Então, cada viewmodel herda aqui dessa classe, BaseViewModel. BaseViewModel, a gente definiu como sendo uma outra classe.

[02:16] Então, aqui em BaseViewModel, a gente definiu esse método OnPropertyChanged, que vai fazer o quê? Ele vai invocar esse OnPropertyChanged, que é esse (render) de eventos, para poder notificar a view de que houve uma mudança numa propriedade.

[02:34] Então, sem isso, a nossa view não sabe que teve uma mudança, então ele não se altera, não se atualiza conforme você muda uma propriedade. Agora, se você chama a OnPropertyChanged, a view se atualiza automaticamente. Então, ela buscou os dados mais recentes para poder exibir uma visualização atualizada para o nosso usuário.

[02:59] Então, isso é o padrão MVVM, o que mais a gente viu? A gente viu também como fazer troca de mensagens entre componentes do Xamarin Forms, para evitar esse acoplamento. Então, invés de você chamar uma classe diretamente, o componente chamar...

[03:20] A model, chama a view, a view chama a model diretamente, isso criaria muito acoplamento. Então para evitar isso, a gente faz o quê? A gente troca mensagens através desse componente MessagingCenter do Xamarin Forms. A gente utilizou isso, por exemplo, quando a gente aqui no listagem viewmodel...

[03:41] Por exemplo, quando a gente seleciona o veículo, então a gente utiliza o MessagingCenter para enviar uma mensagem, para o serviço de mensagem do Xamarin Forms, de que houve aqui o veículo selecionado. Então, um veículo foi selecionado pelo usuário.

[03:59] Agora, a gente não está chamando nenhum componente diretamente aqui, a gente simplesmente lança uma mensagem. Aí, na outra ponta, a gente tem que ter alguém que recebe essa mensagem. Esse outro alguém que recebe a mensagem, é o code behind, a listagem view que vai assinar...

[04:16] Vai fazer a assinatura. Olha só, aqui no MessagingCenter.Subscribe, eu tenho o quê? Eu tenho a assinatura dessa mensagem, para poder fazer o tratamento e poder agir adequadamente, conforme a ação do usuário. Então, isso é o sistema de mensageria Xamarin Forms, é o MessagingCenter.

[04:38] Aí, em seguida, a gente viu o quê? A gente viu ações com o command, então o command é o quê? Ele é o substituto daqueles eventos click do botão. Então, por exemplo, lá no detalhe view, a gente tinha um botão, que é o botão de próximo.

[04:54] Então, invés de você assinar o método com um click, a gente trocou aqui esse click, pelo comando. Então, a gente criou uma propriedade de comando. Então, a gente usou a propriedade de comando do botão, para poder chamar... para fazer o binding, para um command chamado ProximoCommand.

[05:15] Esse ProximoCommand, ele vai estar aonde? Ele vai estar na origem do binding, no contexto de binding dessa view, detalhe view, ou seja, vai estar lá no detalhe viewmodel. Então, o detalhe viewmodel, a gente criou o próximo command.

[05:30] Então, o próximo command, que está aqui embaixo, ele foi definido como um novo comando e quando ele é acionado, o que acontece? A gente está enviando uma mensagem. Então, a gente cria aqui uma action, utilizando uma expressão lambda.

[05:47] Então, aqui a gente tem um método anônimo que vai fazer... vai executar uma ação de acordo com o click desse usuário nesse botão. Então, com isso, a gente substitui aquele click que tem lá no... aquele evento de click que tem lá no code behind, para a gente está passando para um outro componente que é o viewmodel, da view de detalhe.

[06:11] Para fazer esse tratamento do evento do click. Ai, prosseguindo, a gente viu como atacar o problema do isolamento da aplicação. Então, invés de a gente ter uma lista fixa de veículos para o usuário selecionado... a gente não quer isso, porque cada vez que uma listagem é alterada.

[06:34] A gente teria que criar uma nova versão da aplicação, mandar para as lojas, distribuir para as lojas e os usuários teriam que baixar esses aplicativos novos, para cada uma das pequenas alterações que a gente tivesse nessa lista. Então, a gente pode fazer o quê?

[06:48] A gente pode fazer a nossa aplicação, quebrar esse isolamento, se conectar no servidor externo da Alura Car e através de um serviço Http GET, a gente buscar a lista atualizada de veículos, cada vez que o usuário iniciar a aplicação, isso que a gente fez.

[07:07] A gente fez isso lá na listagem viewmodel, a gente utilizou aqui essa URL da Alura Car, para poder fazer o acesso ao serviço Http GET. Então, a gente fez o quê? A gente inicializou aqui, criou uma instância do componente HttpClient, do .NET. Esse componente permite que a gente faça essa requisição get.

[07:37] Então, essa requisição get é assíncrona. Então, a gente pegou esse resultado a partir da URL que fornece a lista de veículos, a gente transformou isso num... através do Jason, a gente transformou isso numa listagem, numa lista do (dot net).

[07:58] E a gente conseguiu pegar cada um dos elementos dessa lista, que veio lá do servidor da Alura Car e transformou isso numa listagem que pode ser exibida na aplicação, populando aqui a listagem de veículos da listagem view. Então, a gente fez isso na viewmodel.

[08:17] E automaticamente a view se atualizou, então ela pegou a lista mais recente que veio lá do servidor da Alura Car. Aí, prosseguindo, a gente fez o quê? A gente fez o caminho ao contrário, a gente, invés de pegar os dados da Alura Car, a gente enviou os dados lá para o servidor da Alura Car, através do serviço Http POST.

[08:38] Então, a gente fez isso em que momento? Lá quando a gente termina o agendamento, no que a gente salva o agendamento, olha só, a gente tem aqui um método de salvar agendamento... A gente fez alguma coisa similar, a gente instanciou o novo componente HttpCliente.

[08:54] Então, esse cliente foi utilizado para a gente criar aqui um novo Json, um novo objeto para envio dos dados do agendamento do usuário e a gente utilizou o método PostAsync para poder, de modo assíncrono, enviar para o servidor da Alura Car os dados e conseguir finalmente salvar o agendamento lá no servidor.

[09:22] Então, a gente fez o quê? A gente quebrou o isolamento da aplicação, a gente fez a nossa aplicação conversar com o mundo, para poder realmente ter uma utilizada lá para a Alura Car e eles terem esses dados atualizados em tempo real. Então, ficamos por aqui.

[09:43] Essa nossa aplicação vai ter continuidade na parte três e quatro do nosso curso que vão vir a seguir. Então, a gente fica aqui com a parte 02. Espero que vocês tenham gostado. Façam os exercícios, estudem bastante e aguardem, porque logo a gente vai ter a parte 03 e 04 do curso.

[10:03] Muito obrigado, então. Até a próxima.