09

AgendamentoViewModel e revisão

Transcrição

[00:00] Agora que a gente já tem um ViewModel para a página de listagem de veículo e de detalhe de veículo, a gente também tem que criar um ViewModel para a página de agendamento de teste drive. Então, a gente vai fazer isso, criando uma nova classe, chamada: AgendamentoViewModel. Então, eu vou na nossa pasta de ViewModels.

[00:23] Vou adicionar uma nova classe, então coloco AgendamentoViewModel e vou deixar ela pública. Agora, o que essa classe vai receber quando ela for instanciada? Ela vai ter que receber o veículo que foi selecionado lá atrás na página de detalhes. Então, eu tenho que criar um construtor e esse construtor vai receber um veículo.

[00:55] Veiculo e aqui eu passo o veículo. Agora, eu preciso preencher essa classe com as propriedades que são adequadas para um ViewModel. Então, eu vou voltar no nosso code behind do AgendamentoView e vou pegar tudo o que for referente à ViewModel. Então, vamos descer, vou pegar desde o agendamento, até a última propriedade, antes do construtor da nossa view.

[01:34] E vou mover essas propriedades lá para o nosso ViewModel. Vou colocar aqui em cima. Então, agora o que eu vou fazer? Eu vou ter que modificar o nosso construtor do ViewModel, para ele saber, para ele instanciar quem é o nosso agendamento, quem é o modelo do agendamento que ele vai utilizar para preencher os dados.

[01:59] Então, eu vou remover essas linhas do construtor, do AgendamentoView e vou passar para o ViewModel. Agora, eu preciso modificar o binding context dessa página, como que eu faço isso? Eu tenho que ir lá no nosso code behind da página e trocar o binding context, invés de ser uma instância do próprio code behind, que é o this.

[02:25] Eu vou colocar aqui new AgendamentoViewModel e vou passar aqui dentro, no construtor, eu vou passar o veículo que está sendo recebido por essa página. Agora, a gente vai rodar a aplicação e ver como que se comporta a página de agendamento com o ViewModel. Então, opa, deu erro aqui.

[02:53] Então, vamos ver, ele está reclamando porque o veículo, ele não existe mais aqui na nossa classe do code behind, então aqui nessa referência, ele não consegue encontrar o veículo e não consegue... Não vai conseguir referenciar aqui a propriedade nome. Então, a gente vai ter que corrigir isso agora.

[03:16] Então, o que a gente pode fazer? A gente pode criar uma nova propriedade, eu vou chamar de... Eu vou colocar aqui como tipo: AgendamentoViewModel, e esse AgendamentoViewModel, eu vou colocar como propriedade no nosso code behind. Então, eu vou chamar ela de ViewModel.

[03:36] Então, invés de instanciar diretamente no binding context, eu vou criar numa linha antes, eu vou colocar this.ViewModel = new AgendamentoViewModel e passando o veículo. Então, com isso, eu consigo preencher essa propriedade. E aí, no binding context, eu passo o this.ViewModel, como referência para ele saber quem é o binding context dele.

[04:06] Agora, nessa última linha, eu vou ter que referenciar o ViewModel.Agendamento.Veiculo.Nome e vou fazer isso também para as outras propriedades. Então, aqui, eu vou colocar ViewModel.Agendamento.Nome, .fone, .Email, data do agendamento e hora do agendamento, deixar aqui alinhado.

[04:43] Agora, a gente vai rodar a aplicação e ver como fica a última página, a página de agendamento. Agora, rodando a aplicação, vamos ver. Selecionei um veículo, próximo. Agora, a gente tem aqui a nossa página de agendamento, que a gente vai preencher, para ver se está tudo funcionando corretamente.

[05:07] Então, o nome: fulano de tal; vou colocar aqui o telefone: 1234-5678; aqui o Email: fulano@gmail.com, ok. Aqui, data e hora, não vou mexer e vou agendar. Aí, aparece veículo: HB20 S, o nome: fulano de tal, telefone, Email, data e hora do agendamento corretos. Então, com isso, a gente acabou de ver como funciona o MVVM.

[05:39] A gente fez a parte principal que são os binding. Então, a gente conseguiu fazer essa refatoração, para remover do code behind todo o código, todo aquele código que é referente à lógica de apresentação. E olha só, a gente conseguiu mover do ListagemView, do DatalheView também. Olha só, como ficou muito mais enxuto o nosso código.

[06:07] E no AgendamentoView, também a gente conseguiu remover muitas linhas de código e passar isso para uma outra classe ou melhor, para outras classes, que vão fazer a lógica de apresentação para a gente. Então, vamos fazer uma revisão do que a gente viu hoje.

[06:25] O que a gente viu hoje foi o padrão Model – View – ViewModel, olha só, Model View ViewModel. Então, a gente tem aqui o M do Model, o View que é o V e o ViewModel, que é o VM. Então, Model View ViewModel. Então, no começo dessa aula, a gente viu como remover da nossa view, para o model.

[06:50] A gente moveu a lógica e os dados de negócio, a gente fez isso com o veículo e a gente fez isso também com a listagem, aquela lista que tem alguns veículos lá preenchidos, a gente moveu tudo isso lá para o nosso model. Depois, a gente começou a ver o ViewModel. Então, a gente começou a passar do nosso view para o ViewModel as notificações e o data binding.

[07:15] Então, notificações, data binding, a gente já viu. A gente viu também que para fazer as notificações e o data binding, a gente tem que instanciar uma nova ViewModel, a cada página, a cada view que a gente trabalha. E a gente viu também o método chamado OnPropertyChanged, que precisa ser implementado para poder fazer essa notificação das propriedades, dentro de um ViewModel.

[07:48] Então, mais para frente, a gente vai ver como fazer aqui os comandos. O comandos, a gente não viu ainda, que são algumas ações, que são executadas quando, por exemplo, um usuário seleciona um elemento de uma lista ou quando ele clica num botão, que é a principal função de um comando.

[08:07] Então, espero que vocês tenham gostado da aula, muito obrigado. Até a próxima!