02

Acoplamento model

Transcrição

[00:00] Oi, pessoal, tudo bem com vocês? Eu sou Marcelo Oliveira. Bem-vindos de volta ao nosso curso do Xamarin com Visual Studio. Pelo o que a gente viu no curso até agora, toda a nossa lógica da aplicação está contida dentro das views. Eu tenho aqui três views.

[00:18] Eu tenho a view de listagem, a view detalhe e a view agendamento. Então, todo o nosso código, toda a nossa lógica, seja lógica de negócios ou lógica de apresentação, está toda ela contida aqui dentro das nossas views. Cada view tem um code behind. Então, olha só, eu não tenho nada fora das views.

[00:41] Então, essa estrutura pode ser um pouco problemática para a gente, porque a gente está misturando junto com a apresentação, a gente está misturando a lógica de negócios e a lógica de apresentação também. Então, olha só como está a estrutura atual da nossa aplicação.

[01:02] Aqui nesse desenho, a gente vê, que a gente tem aqui um desenho de uma view. Então aqui nessa parte azul, a gente tem a apresentação dela feita com o xaml e por trás dessa view, a gente tem o code behind, que é o código que está atrás da nossa página xaml.

[01:28] Então esse código que está aqui pontilhado, ele está em C Sharp. Então, esse código C Sharp, ele está fazendo a lógica de apresentação, ele está fazendo também o data binding com o xaml, está trabalhando com o xaml para fazer o binding das propriedades e está fazendo também a lógica de dados e negócios.

[01:56] Agora, o que a gente precisa fazer para melhorar isso? Porque a gente está tendo muito acoplamento aqui, a gente está misturando muitos conceitos dentro de uma view e a view não deveria servir para isso. A view serve para simplesmente mostrar os dados para o usuário.

[02:16] E estar ali presente para poder receber o input, para poder receber a interação do usuário. Então, o primeiro passo para a gente melhorar essa estrutura, essa arquitetura da nossa aplicação é pegar aqui a lógica de dados de negócios e mover para outro lugar mais adequado e remover ele do nosso code behind.

[02:43] E assim, a gente vai livrar a nossa página de um trabalho que não deveria ser dela. Então, a gente vai fazer isso agora. O primeiro passo é verificar que na nossa listagem view, a gente tem aqui, além da partial class, que é a listagem view, a gente também tem a class veículo.

[03:04] Então, não é bom deixar duas classes no mesmo arquivo. O que a gente vai fazer agora é pegar esse veículo, que é o nosso modelo e mover ele para uma outra pasta chamada models. Então, a gente já tem uma pasta chamada views, eu vou criar aqui uma pasta chamada models.

[03:24] Então, eu criei aqui models e dentro de models, eu vou ter os modelos que vão existir na nossa aplicação. Então, dentro de models, eu venho aqui em adicionar em adicionar classe e vou dar o nome dessa classe de veículo, que vai conter a lógica do veículo da nossa aplicação.

[03:46] Eu vou copiar aqui, simplesmente recortar essa classe pública veículo e colar aqui na pasta de models. Agora, a aplicação vai começar a falhar, porque a classe veículo mudou de namespace. Agora, ela não é mais TestDrive.Views. Ela é uma TestDrive.Models, como a gente está vendo aqui.

[04:12] Então o Visual Studio está começando a reclamar aqui de algumas referências a essa classe veículo, então eu vou acertar as referências com o Ctrl + ponto + ponto, troco para using TestDriveModels. Já resolvi a referência. Ele está reclamando agora aqui desse veículo, mas é porque o veículo que está sendo passado para dentro do detalhe view.

[04:42] Lá no detalhe view, a gente tem uma referência também com o namespace antigo. Então, eu vou trocar aqui, vou colocar Ctrl + ponto + ponto e troco para using TestDriveModels. Acertei a referência. Agora, aqui tem um outro problema na referência, que a gente passa para o construtor do agendamento view, aqui em baixo.

[05:05] Vamos lá com F12, vou navegar para o agendamento view, vou acertar a referência com Ctrl + ponto + ponto, using TesteDriveModels. Agora deve funcionar. Vamos rodar aplicação, ver se não quebrou nada. Agora vai rodar o emulador, está rodando o emulador agora e vai rodar a aplicação, vamos ver.

[05:32] Legal, apareceu aqui a listagem dos veículos, então a gente trocou o nosso modelo, a gente trocou de lugar, colocou na pasta models, trocou os namespaces, está funcionando normalmente. Agora, uma outra melhoria que a gente também pode fazer, é mover a nossa lista de veículos para uma outra classe...

[05:56] Para um outro modelo. Olha só, a gente está fazendo aqui, dentro do nosso ListagemView.xaml.cs, que é o code behind, a gente está colocando a listagem de veículos. Só que a gente poderia fazer melhor, se a gente movesse isso para uma outra classe, especializada dentro do modelo.

[06:17] Então a gente vai criar uma classe aqui dentro de models e vou chamar de listagem veículos, que vai conter a lista de veículos. Então, aqui dentro eu vou deixar essa classe pública e no construtor, eu vou preencher essa classe, vou preencher uma lista aqui dentro com a lista que já existe dentro do nosso code behind.

[06:50] Então aqui no code behind, eu vou remover essa atribuição dessa lista de veículos, vou recortar e vou mover lá para a nossa classe, a listagem veículos. E agora, o nosso code behind, eu vou pegar e acessar essa listagem veículos para “popular” a nossa listagem de veículos, lá dentro do code behind.

[07:12] Então eu coloco aqui this.veiculos = new listagemVeciulos().Veiculos. Então, a agora a gente vai rodar a aplicação, ver se não quebrou nada, se está funcionando ainda. Agora, está rolando o emulador, vou rodar a aplicação. Legal, está aparecendo aqui a lista sem problemas.

[07:37] Então, a gente conseguiu remover a definição dessa listagem. A gente moveu isso lá para o nosso modelo, para tirar da nossa view a responsabilidade de criar uma lista de veículos. Então, como fica o nosso code behind? A gente continua tendo a propriedade veículos.

[07:59] Só que agora, a gente não está instanciando uma listagem de veículos. Então, com isso a gente tem um code behind mais enxuto. Então, continuando com essa nossa refatoração, a gente pode ir para a próxima view, que é a view de detalhes e ver se a gente pode extrair alguma coisa para o modelo.

[08:19] Estou entrando aqui em detalhe view, do lado direito da tela e no detalhe view, logo de cara a gente vê que tem algumas constantes aqui, que representam o preço de cada um dos acessórios, mas olha só, detalhe view serve para quê? Serve para exibir simplesmente a nossa tela de seleção de acessórios do veículo.

[08:43] Só que essa tela, sendo uma tela de exibição de dados, ela não deveria se preocupar, por exemplo, com o preço do freio ABS ou preço do ar-condicionado ou preço do MP3 Player. Então, essas constantes, elas fariam mais sentido, se elas estivessem numa classe especializada em tratar do veículo.

[09:04] Então, olhando o nosso modelo, a gente tem essa classe já, que é a classe veículo. Então, o que a gente vai fazer agora? É mover essas três constantes lá para dentro da nossa classe veículo. Eu estou recortando aqui, recortei e vou colar lá em veículo. Só que se eu removo essas três constantes, o que acontece?

[09:30] Eu vou quebrar a minha classe, o meu code behind e do detalhe view. Então, eu vou ter que acertar algumas coisas aqui, para fazer ele voltar a funcionar. Então, olha só, o Visual Studio começou a reclamar aqui, que ele não está encontrando a referência para o freio ABS.

[09:48] Claro, porque a gente moveu essas três constantes lá para a outra classe. Então, eu vou ter que, invés de simplesmente jogar aqui o frio ABS, eu tenho que colocar primeiro uma referência do objeto que contém o freio ABS, que é a instância veículo. Então, eu coloco Veiculo.FREIO_ABS.

[10:09] Só que com isso, eu continuo tendo problemas, por quê? Porque freio ABS, ele está marcado como privado, então eu teria que marcar isso como público. Então, eu vou modificar aqui de privado para público, vou fazer isso com as três constantes. Então agora está resolvido, pelo menos para o freio ABS.

[10:35] Então, eu vou fazer isso para as outras constantes. Então, aqui embaixo, ar-condicionado, eu coloco Veiculo.AR_CONDICIONADO, Veiculo.MP3_PLAYER. Aqui em baixo, quando a gente está exibindo a mensagem para o usuário, a gente tem acessa essas três constantes.

[10:54] Porque a gente tem que verificar qual é o preço de cada um dos acessórios. Então eu vou colocar aqui: Veiculo.FREIO_ABS, Veiculo.AR_CONDICIONADO e Veiculo MP3_PLAYER. Agora a gente roda a aplicação e vê se não quebrou nada. Agora está rodando o emulador, vai rodar a aplicação.

[11:17] Então agora eu consigo verificar aqui os preços de cada um dos acessórios, consigo marcar e desmarcar. Então, chaveando aqui, a gente vê que o preço está sendo alterado com sucesso aqui em baixo.