07

List, ObservableCollection e ActivityIndicator

Transcrição

[00:00] Agora a nossa aplicação conseguiu acessar o serviço http rest com sucesso, como a gente pode ver aqui. Então, primeiro a gente utilizou a biblioteca Microsoft Net Http para poder criar uma instância dessa classe HttpClient. A gente usou esse client para fazer a chamada para o GetStringAsync.

[00:25] E aí, pegando esse resultado, que é uma string, como a gente pode ver aqui, a gente converteu essa string numa rede de objetos do (dot net). A gente fez isso utilizando a biblioteca Newtonsoft, mais exatamente, a gente utilizou a classe JsonConvert, que vai converter a string para o nosso array de objetos.

[00:49] O que está faltando agora, quando a gente executa a aplicação, a gente vê que a lista ainda não apareceu na tela, porque a gente foi lá no serviço, chamou o serviço, mas não fez nada com os dados. Então, essa listagem, é lógico, ela não vai aparecer na tela.

[01:07] Então, o que a gente precisa fazer é popular essa lista, essa lista foi inicializada aqui em cima, this.Veiculos = new List de veículos. Só que essa lista está vazia, ela precisa ser inicializada, ela precisa ser populada. Então, a gente vai fazer aqui um (laço) foreach.

[01:28] E a gente vai... para cada elemento do nosso resultado, a gente vai fazer foreach para cada veículo, veiculoJson. Dentro de veiculosJson. Então, a gente vai adicionar um elemento novo na nossa lista de veículos. Então, eu faço this.veiculos.Add(), para adicionar e aqui dentro, eu vou adicionar o novo veículo.

[02:00] Então, eu coloco new Veiculo. Então, aqui dentro da chaves, eu coloco as propriedades que vão ser populadas. Como a gente só tem duas propriedades no retorno do serviço, que é o nome e o preço. Então, eu só vou colocar aqui dentro nome, que vai vir do veiculoJson.nome,.

[02:20] E o preço, que vai ser veiculoJson.preco,. Então, com isso, a gente vai conseguir alimentar essa lista, a nossa lista que inicialmente estava vazia. Vamos rodar a aplicação e ver essa lista ser populada. Então, agora ele vai rodar a aplicação. Rodando agora. Legal, então ele parou aqui no nosso breakpoint.

[02:49] Vamos ver o que ele adicionou. Então, agora this.Veiculos tem 13 elementos. Então, a gente conseguiu com sucesso, alimentar... olha, a gente conseguiu popular a nossa listagem, que inicialmente estava vazia. Então, agora a gente uma lista de veículos que está sendo atualizada com o resultado da requisição para o serviço externo.

[03:13] Então, agora a gente vai rodar a aplicação e ver essa lista aparecer na tela. Opa, não apareceu nada, isso é muito estranho. Por que será que não aconteceu nada? Bom, a gente tem alguns problemas aqui, que é o seguinte. A gente está trabalhando com um tipo de lista, que é o list.

[03:35] E o list é o seguinte, ele é uma lista que é obtida inicialmente pela aplicação, quando você está iniciando a página. Então, quando a gente faz o binding lá do view, para o viewmodel, a gente faz o binding. E aí, olha só, ele passou por aqui e viu que veículos é uma lista vazia.

[04:00] Então, quando ele fez o binding no nosso xaml, vamos olhar aqui o xaml. Então, aqui dentro do xaml, a gente tem listagem view, a gente tem o ListView e aqui o ItemsSource, que é o Binding lá para a nossa propriedade veículos, que é um list do objeto veículo, que está lá no viewmodel.

[04:23] O problema é que quando esse binding é feito, o que acontece? A nossa lista está vazia. Então, nesse momento, aqueles dados são apresentados e como essa lista é vazia, não aparece nada, claro. Só que num outro momento, mais para frente, é que a nossa página vai realmente buscar do serviço os dados para adicionar na lista.

[04:47] O problema é que o list está sendo populado, só que lá, a nossa view, ela não sabe que o list foi alterado, então isso causa esse problema. Então, como a view não sabe que o list foi alterado. Então, ela não apresenta os novos veículos que vieram a partir do serviço.

[05:08] Então, o que acontece? A gente tem que utilizar um outro tipo de objeto, um outro tipo de classe que vai realmente notificar a view. Vai falar assim: "Olha, view, eu quero que você exiba também esses dados, porque eu fui atualizado.". Então, a nossa lista de veículos, que é a do tipo list, ele não tem essa capacidade.

[05:33] Então, a gente tem que utilizar uma outra classe mais especializada, que possa ser observada pela view. Então, ela tem que ser observável, tem que ser uma coleção observável. E o que é uma coleção observável? Em inglês, observable collection. Então, a gente vai entrar aqui na definição de veículos...

[05:55] E vai trocar a list por ObservableCollection. Então, ele não achou a referência, então, eu faço Ctrl + . + ., vou colocar aqui o using System.Collections.ObjectModel. Agora, ele achou a referência. Então, eu vou ter que trocar também aqui em baixo.

[06:20] Quando ele está instanciando essa lista, eu vou ter que colocar new observableCollecion de veículo. Então, com isso, a gente está trocando o tipo da lista, para que a nossa view saiba, para que ela possa observar a nossa lista e saber se teve uma mudança.

[06:41] Então, a gente vai rodar a aplicação e ver como que ela se comporta agora. Agora está rodando a aplicação. Muito bem, agora ele fez a chamada para o serviço. Ele converteu aquela string para uma array de objetos. Esse array de objetos foi usado para popular a nossa lista, que agora é um ObservableCollection.

[07:05] E aí, com o ObservableCollection, a gente conseguiu com sucesso exibir na tela... Olha só, a gente tem aqui vários elementos que vieram lá do nosso web service. Então, com isso, a gente conseguiu fazer o ciclo completo, para fazer essa chamada para o serviço http rest.

[07:26] Mas a gente ainda tem um outro problema, que é o seguinte, você viu que no início dessa chamada, quando a aplicação está iniciando, ainda tem um tempo de resposta, é um pouco longo. Então, a gente vai ter que fazer o quê? Olha só, ele vai iniciar a aplicação.

[07:44] Então, olha só, a aplicação está rodando, só que ele levou um tempo ainda para exibir os dados. Então, se o serviço for demorado, se a conexão do cliente, do nosso usuário estiver lenta, se ele estiver num lugar que está com difícil acesso, o que acontece?

[08:03] Ele pode esperar muito, por muito tempo e achar que está com algum problema na aplicação, ele pode achar que a aplicação está travada, ele pode achar que teve um erro que ele digitou alguma coisa errada. Então, o que acontece?

[08:18] A gente tem que mostrar para o usuário que ele tem que esperar, aguardar o serviço completar para ele poder usar o aplicativo. Então, o que a gente vai fazer agora, é exibir para ele uma notificação, um pequeno recurso visual, que vai permitir que ele espere...

[08:38] E sem ele ficar preocupado, se a aplicação travou ou não. Então para isso, a gente vai parar a aplicação. Agora a gente vai lá para o xaml, lá para view, que é a listagem view e a gente vai adicionar aqui um novo componente e esse novo componente é um indicador de atividade ou do inglês activity indicator.

[09:15] Nessa página, a gente está colocando aqui dois controles. Então, para a gente não ter problema com relação ao layout, a gente vai simplesmente colocar dentro de um StackLayout, como a gente já aprendeu antes. Então, a gente coloca dentro de um StackLayout.

[09:33] E aí ficam os dois controles dentro do StackLayout. E o ActivityIndicator vai aparecer aí na parte de cima. Agora, o que eu vou fazer? Eu vou mover essa margem lá para cima, essa margem vai ficar aqui no StackLayout. E agora, o activity indicator vai ter uma propriedade que vai indicar se a aplicação está ocupada.

[10:00] Então essa propriedade, se ela está rodando, se a aplicação está rodando. Então, IsRunning, quer dizer, se a aplicação está rodando ou não. Então, se eu colocar aqui, por exemplo, true. O que vai acontecer? A gente vai ter indicação de que a atividade, a aplicação está rodando.

[10:20] Então, vamos ver, estou rodando agora. Então, olha só, está aparecendo aqui o indicador de atividade, dizendo que a aplicação está rodando e continua rodando. Opa, deu erro no serviço. Eu vou rodar de novo, estou rodando. Legal, agora vai abrir a nossa página e vai iniciar a aplicação.

[10:48] Olha só, então apareceu esse símbolo aqui girando, que é próprio do Android, cada dispositivo, cada sistema operacional tem o seu. O Android é essa argolinha aqui girando. Então, ela vai indicar para a gente que a aplicação está ocupada, só que agora já apareceu aqui a listagem com os veículos.

[11:11] Então, o serviço já respondeu a nossa requisição e já exibiu, a gente já conseguiu exibir aqui a listagem para o usuário. O problema é que essa indicação, esse indicador de atividade, ele não para, ele está continuamente exibindo para o usuário.

[11:27] Então, a gente tem que utilizar alguma estratégia para que esse indicador, ele só apareça quando o serviço está sendo chamado e quando os serviço responde, ele vai ter que ser parado, vai ter que sumir da tela, para o usuário saber que já completou.

[11:46] Então, a gente vai criar uma nova propriedade lá no nosso viewmodel, que vai se chamar aguarda. Então, vamos entrar aqui no viewmodel, vou colocar aqui uma propriedade, vou colocar propfull, para ele criar automaticamente tudo o que precisa criar na propriedade. Já criou.

[12:10] Eu vou chamar... eu vou colocar o tipo dessa propriedade como “booleano”, ou seja, está aguardando ou não, está ocupado ou não. Então, ela vai ser um tipo “booleano”, aqui em baixo também boolean. Eu vou trocar o nome MyProperty, eu vou chamar de aguarde.

[12:32] E essa variável interna, eu vou chamar de aguarde, minúsculo. Então, agora, além disso, eu tenho que fazer uma outra modificação aqui nessa propriedade, porque como a gente já viu no MVVM, quando a gente altera o valor de alguma propriedade, a gente tem que notificar a view.

[12:59] Porque, senão, o binding, ele não sabe que houve alteração na propriedade e ele não sabe também como se atualizar. Então, para isso, a gente vai ter que utilizar o nosso on property changed. Mas olha só, a gente não tem on property changed nessa viewmodel.

[13:21] A gente implementou esse on property changed em outra viewmodel, que é a de detalhe. Então, olha só, lá no detalhe viewmodel, a gente tem a implementação do INotityPropertyChanged, que é a interface que vai permitir que a gente coloque esse método que notifica a view que a propriedade foi alterada.

[13:48] Então, a gente tem aqui a implementação dessa interface e tem esse código que é feito exclusivamente para isso. Agora, eu poderia simplesmente copiar esse código e colocar na outra viewmodel, que é a viewmodel de listagem. Só que invés disso, o que eu vou fazer?

[14:07] Eu vou fazer uma coisa mais inteligente. Eu vou criar uma criar uma viewmodel base e eu vou herdar, fazer uma herança dessa viewmodel, para que eu possa reaproveitar código, que eu não precise mais ficar copiando código de um lado para o outro. Então, vamos lá.

[14:29] Eu vou criar aqui uma base viewmodel, a base viewmodel, vou criar uma classe na pasta viewmodel e vou chamar de base viewmodel. Essa classe, ela vai se pública, só que ela vai ter que ser abstrata, porque ninguém vai poder usar uma instância de base viewmodel.

[14:58] Você só pode utilizar uma herança, alguém que herda de base viewmodel, então eu coloco abstract para isso mesmo. Agora, eu coloco em base viewmodel as propriedades que vão ser reutilizadas em outras classes. Então, eu vou copiar esses campos aqui, essa propriedade e o evento.

[15:23] Copiei para o base viewmodel e aí, eu vou herdar, invés de colocar aqui que essa classe, detalhe viewmodel, ela implementa uma interface, INotityPropertyChanged, eu vou colocar que ela herda de base viewmodel. Então, agora eu vou modificar a nossa classe.

[15:51] Base viewmodel, para ela implementar interface INotityPropertyChanged. Pronto, já fiz isso e agora vou ter que resolver esse nome aqui no meio também. Então, agora tudo certo. Eu vou ter que fazer o mesmo para a nossa classe de listagem viewmodel, que ela vai ter que herdar de base viewmodel também.

[16:18] Agora, a listagem viewmodel vai herdar de base viewmodel, e aí, eu vou poder com sucesso acessar INotityPropertyChanged, porque aí, eu consigo notificar a view de quando a propriedade aguarda for modificada. Então, o próximo passo agora é a gente utilizar a propriedade aguarda como binding para a propriedade do nosso indicador de atividade do activity indicator.

[16:53] Então, a gente vai lá na view, na listagem view e modifica aqui onde está o IsRunning do ActivityIndicator, a gente troca por um binding, que vai amarrar, que vai se amarrar na propriedade aguarde do viewmodel. Além disso, eu também vou fazer aqui o IsVisible, vou colocar o IsVisible com o binding no aguarde também.

[17:25] Porque, mesmo que o indicador de atividade não esteja rodando, ele também ocupa espaço. Então, quando ele não estiver rodando, eu também quero que ele seja invisível. Então, eu coloco aguarda aqui como binding, beleza? E a gente vai rodar a aplicação e ver como ela se comporta.

[17:49] Então, agora vai rodar a aplicação, rodando. O problema é que agora não apareceu, nem apareceu o indicador, por quê? Bom, porque a propriedade aguarde, ela é um “booleano”, ele é um verdadeiro ou falso, só que o valor inicial, o valor (de fo) desse verdadeiro ou falso é falso.

[18:12] Ou seja, durante todo o tempo, a nossa propriedade aguarde está como falsa, então, para a nossa view, a propriedade aguarda, como está falsa, então ela nem apresenta o indicador de atividade. Então, o que a gente precisa fazer é iniciar o indicador de atividade com o true...

[18:32] No momento em que o serviço é chamado, porque, aí, sim, a gente tem uma atividade que vai consumir um certo tempo. Então, vamos lá. Agora, eu vou modificar a listagem viewmodel, eu vou lá na get veículos e aqui, eu vou colocar o aguarde como true.

[18:58] Então, no início da chamada dos veículos, a partir do serviço http, eu quero que a aplicação aguarde. Então, aguarda, nesse ponto, tem que ser true. E aí, quando a aplicação terminar, quando ela responder, eu quero que o aguarde seja falso.

[19:18] Então, lá o final do get veículos, eu quero colocar aguarde é igual a false. Vamos rodar agora a aplicação e ver o que acontece. Então, agora vai rodar a aplicação, está carregando o indicador. Muito bem, então ele carregou, ele mostrou o indicador de atividade, pediu para o usuário esperar.

[19:44] E aí, quando os dados foram carregados, ele mostrou a lista na tela e fez o indicador de atividades sumir da tela, que é o comportamento esperado. Então, com isso a gente conseguiu fechar esse ciclo de fazer uma requisição para o serviço http, esperar ele responder, exibir para o usuário esse feedback...

[20:07] Essa notificação de que tem alguma coisa que pode demorar, que está acontecendo, pedindo para ele esperar. E aí, quando o serviço http rest responde através do get, a gente faz o indicador de atividades sumir e a gente exibe na tela para o usuário.