VIDEO: https://www.youtube.com/watch?v=ddmZ6k1GIkM&list=PLdo4fOcmZ0oUBAdL2NwBpDs32zwGqb9DY&index=6

|  |  |
| --- | --- |
| **ENGLISH** | **INDONESIA** |
| We're continuing our journey learning.NET MAUI to build beautiful native cross-platform desktop and mobile applications.  Now, so far, we've built out an application with XAML, an XML-based markup for user interface, we've then extended it with MVVM and data binding to make our application responsive.  Now what we're going to do is something very common built into every single application, which is navigation.  That's right, navigating from page A to page B. Additionally, what's great is that we can pass data between our pages too.  Now, built into .NET MAUI are several ways of doing navigation, specifically, we're going to take a look at URI shell-based navigation.  I talked about the shell in an earlier video giving the structure to our application, but what's great about shell is that it not only includes a built-in dependency service to do constructor injection and dependency injection but it also enables us to do URI-based navigation.  A URI, similar to a website. You go to slash details and it will go to the details page. You can also send query parameters that can be simple data types or complex objects as well.  It's very easy to do with .NET MAUI with just a few lines of code. Let's hop over to Visual Studio 2022 and get started. So far, we started to build our My Tasks application. Here, we have a full XAML user interface and data bindings.  I can add different items inside of here, and get nice animations. I can even swipe to delete. Now, this is really great, but we want to also be able to maybe tap into an item and go to details.  What we're going to do now is add navigation to our application.  Let's go ahead and do that. The first thing here is that this is the same application that we've been building through this entire beginner series.  We have our Main Page, we have our Image, our Entry, our Button, and our CollectionView here.  We've added a SwipeView so we can add that swipe to delete.  We also have the MainViewModel, which is essentially the code behind that page that's doing different observable properties that we can do data binding to.  So that text property that we see there has a data binding to this text property right here on the Entry.  Let's add another page that we can navigate to. I'm going to go ahead and I'm going to right-click on the MauiApp2 and say Add, New Item.  When I do that I'll be brought to this screen that enables me to select different items, and there is even a subcategory specifically for .NET MAUI.  Here we have different pages and views and resources. I'm going to select the ContentPage XAML for .NET MAUI, and we're going to call this DetailPage.  Now, this page is a very simple page that just says "Welcome to .NET MAUI."  There it is. Perfect. There's of course no code behind it, just very simple. Now, the other thing we're going to do is pair this up with a view model.  I'm going to come in and say Add. I'm going to say New Item in here, and under Code, we're going to add a class, and we're going to say DetailViewModel.  I like to name my view models very similar to the page, so DetailPage, DetailViewModel. I've added a new class, so we're going to add some nice C# 10 features here, like file-scoped namespaces. We're also going to make this a public partial class.  We're going to also inherit from our ObservableObject. That observable object is coming from our MVVM toolkit for .NET and MVVM.  That community toolkit gives us everything we need to start doing MVVM here, which is really great.  Next up, what we need to do is we need to associate this DetailViewModel with our page.  Inside of our DetailPage.xaml code behind, what we're going to do is in the constructor say DetailViewModel vm, make sure we come in and bring in that namespace, and then we're going to set the binding context here over to the vm.  Perfect. Now, just like we did for our MainPage and our MainViewModel, we're also going to register those in our MAUI program.  This is going to register them with our dependency service so when we navigate to them, it automatically will associate the DetailPage with  the DetailViewModel.  Now the difference here is that before we're adding singletons, but really every single time we navigate to a new page, we wanted to create a new page.  We're going to change this to AddTransient. Singleton you can think of it as a global where it creates one copy of it and then it will go ahead and keep that memory, else transient will be created and then destroyed and a new one will be created every single time.  This is what we want to have to happen every time we navigate. Now what we're going to do at this point is register the DetailPage with the the routing system of.MAUIShell.  Here inside of our AppShell, which again if you remember here has ShellContent of the MainPage.  The MainPage has a route of MainPage, which is very convenient.  I like to name my routes the same as the MainPage here.  We're going to come in and register this page in the code behind since it's not part of the shell.  If I had tabs or I had to fly out, it'd be part of it, but here we just want to navigate to it.  Let's go ahead and do that. What we're going to do now is we're going to say Routing.RegisterRoute and this is going to take in two things.  First is the route itself, so I could say DetailPage. I can give it the DetailPage. It could be detailed. It could be whatever I want the string to be.  I like to use the name of the property a lot because in C# if we refactor, it will also then cascade down.  Then we're going to associate with a type, so we're going to say typeofDetailPage.  There we go. This is really nice.  This gives me a string of DetailPage to navigate to and a type of DetailPage to navigate to as well.  Perfect. Now let's do some navigation before we actually pass some data across. Here's what we're going to do.  We're going to come into our main page, and there are several ways that we can add navigation when we click on an item in the CollectionView.  Here we can say selected and we can see that there's a selection change event and a selection mode.  That is one way if we want to register an event with the CollectionView, which is an option.  Now, I'm actually going to say SelectionMode is none here because we don't want to use the built-in selection mode.  Because we're just doing navigation, we're not actually selecting anything, I'm more like tapping on an item.  What I want to do here on the Frame itself, which is housed in this label, is I want to add a TapGestureRecognizer.  Another powerful feature is that there are tons of built-in gesture recognizers and one specifically that we'll use is the tap, but there's also a swipe, a pinch, a pan, a drop, and a drag, but here we're going to use a TapGestureRecognizer.  Now, inside here, you can assign a tap to that, or specifically a command, which is very nice, similar to our SwipeView above.  You can also say the number of taps required. If you have something you need to tap 10 times, you could do that as well.  I'm going to go out and simplify this here and I'm just going to go ahead and copy and paste the command from our swipe to delete, and then we're going to change it.  We saw earlier that we needed to get access and say, please do some data binding up the stack to our MainViewModel, and here what we're going to do is instead of going to the DeleteCommand, let's go ahead and say the TapCommand, for example.  Here what we're going to do is pass in a parameter of the string that was bound to the label here.  Now in our MainViewModel, we need to go ahead and add that to that command.  We have the DeleteCommand there, let's go ahead and add another async Task this time of Tap pass in a string of s. There we go.  Now what we're going to do is go ahead and make this an ICommand that again is going to do some source generation for us so it generates the Tap command for us.  Now what we're going to do is we're going to go ahead and do some navigation.  Here I'm going to say await. Again, we're doing this async task because it's actually doing the navigation, and this is going to use the shell navigation, so Shell.Current.GoToAsync.  Now, what I like about this is that I can just pass it in the DetailPage or that route identifier that I defined in my AppShell right here.  I can also pass at the string directly, which is also very nice.  If I wanted to I could come in back into my MainViewModel here and replace this hard-coded string with the name of DetailPage.  Now, this is going to tell the shell to go to that DetailPage directly, which is pretty awesome.    Let's go ahead and compile this up and start actually tapping on an item and then doing navigation to that DetailPage.  Again, this is up to you if you want to reference the DetailPage there or if you want it to be a string or have some association, totally up to you and your application architecture.  Now let's give this a second to compile the application and deploy it. Our application is launching here, and let's go ahead and add an item to our list.  Again, apples, when you add, add that and I'm going to do a tap event, and boom, just like that, we've navigated to a new page and we can see welcome to .Net MAUI.  Now, the other thing I'm able to do is, of course, modify that XAML, so if I want to say font size of 25, for example, I hit "Save" and sure enough that's going to update there.  We can also add a little bit of padding here of let's say 20. Now we get a little padding there, perfect.  Now, that's pretty great that we just did some navigation, but it sure would be great to pass the actual detail page information, so for example, the item I selected, and additionally maybe add a button here that says go back.  That'd be pretty nice. Let's go ahead and do that next.  Here is what we're going to do, is we're going to stop debugging, we're going to go into our main view model again, and now what we're going to do is we're going to pass it information and there are two ways of doing that.  The first way is to add it as a query property, and this is very similar to a URL, so for example, I can go ahead and do some string interpolation here.  Here I'll do some detailed pages and then what I'm going to do is I'm going to say a question mark and then I'm going to say text equals.  That is going to be my query identifier, so that's the ID there. Then I can pass in s for example, which is the string that was tapped  here to go ahead and pass that along.  This is really important because this enables me to send simple data types like strings and integers across.    But what if you had a complex data type, like a person or a monkey or a car or some other data object?  What's great is that you can pass not only through the query property here, but you can also pass in a new dictionary of string object, which is your key values.  For example, here, just go ahead and fills it in, I could for example say, what's the detail page and ask for example, or I could pass in a new object of something, whatever it is inside of here, and you can pass that information across.  If you have complex data types, it works exactly the same as simple data types but you have this big bucket dictionary here, which is really cool.  I'll link to the documentation in the show notes and exactly how to do that as well.  But I'll just go ahead and pass a simple data property across the string.  How are we going to get that string into our detail view model?  Well, we can add a query property in one of two places.  We can do it in our detail page, which would be okay if we didn't have any MVVM and any view models, but we can also add a directly  into our view model itself.  On top of this, we're going to add a query property.  This is really cool and the query property takes in a name, which is going to be the name of the property inside of the detail view model to  bind to so we'll say text, that's what we're going to name it and the query property ID of text as well.  Again, this property here that's passed in is going to be the same exact one that we registered here.  If this was ID, for example, then you change this to ID, just like that.  I like to name them the same. That is just me. It makes it simpler for me to remember. But now we need to create this text property and we're  going to create one that we're going to do some data binding to.  I'm just going to say string text and I'm going to mark that as an observable property.  Again, the reason that this data binding to text will work is that if we go into our dependencies and look at our analyzers for our community  a toolkit that we brought in went ahead and generated those properties for us automatically, so there's the text and it does all that changing for us, which is really nice.  Keeps our code really simple and that's why I absolutely love the .Net Community Toolkit for MVVM.  Now let's add one more thing which, maybe a way to navigate back.  I'm going to do another async task here and say, go back.  There we go. Now what I'm going to do is await on Shell.Current.GoToAsync.  Now to go backward, I don't need to reference the main page because I would actually push a page onto the stack.  I want to navigate back and some of the ways how you would navigate around the file system, I'm going to use the dot, and that is how I'm going  to navigate backward.  You can also do additional navigation at any time by adding a forward slash, so that would navigate back and then to another page, so if I had detail or I had a monkey page, for example, that would navigate to that page, and you could do that a bunch, for example, go up two pages and then also pass parameters back and forth.  It's really nice. Now we're going to do is also mark this as an ICommand because that is going to go ahead and turn it into a command for me and it'll be called Go Back command, which will be automatically source generated.  Now what I'm going to do is go one more time into my detail page, and instead of doing Welcome to MAUI, we're going to do some data binding.  We also need to do our association with our data type, so that way it's compiled and bound. I'm going to say a new xmlns namespace, a view model.  In here, I'm going to type in the view model and it's going to find that it's in this namespace, so we're creating a XAML namespace and I'm going to say x data type and that will be a view model to the detail view model.  Now what I'm able to do is do a binding here to the text property. There it is. Now, let's also create a button here and say text, go back, and then what I'm going to do is register the command to the binding of the Go Back command just like that.  Perfect. Now, let's go ahead and do some data binding again. For all setups see our navigation. What we want to see is whatever item I tap on,  it should go ahead and navigate to that page and then automatically set the text property when I navigate.  Now we have our application up and running. I'm going to say apples were added there. I'm going to add oranges as well. When I tap on one, sure enough, we have apples and a Go Back command that navigates backward.  Oranges go back apples go back. We just implemented navigation to and from pages, back navigation, and the ability to pass query parameters  or full objects using the built-in navigation system of .Net MAUI.  There you have it, you've just done some navigation in your .Net MAUI application. We've passed data from page A to page B, and we figured out how to navigate back as well.  Now, what we're going to do next in the .Net MAUI Beginner Series is take a look at how we can integrate native platform APIs.  Now .NET MAUI gives you them inside of C-Sharp but also gives you an abstraction of our common platform features such as sensors, geo-location, and connectivity.  Stay tuned to the next video in the .Net MAUI Beginner Series to learn more. | Kami melanjutkan perjalanan kami dalam mempelajari NET MAUI untuk membangun aplikasi desktop dan mobile lintas platform yang indah.  Sekarang, sejauh ini, kami telah membangun aplikasi dengan XAML, sebuah markup berbasis XML untuk antarmuka pengguna, kami kemudian memperluasnya dengan MVVM dan pengikatan data untuk membuat aplikasi kita responsif.  Sekarang yang akan kita lakukan adalah sesuatu yang sangat umum yang ada di setiap aplikasi, yaitu navigasi.  Benar, navigasi dari halaman A ke halaman B. Selain itu, yang lebih hebat lagi adalah kita juga dapat melewatkan data antar halaman.  Sekarang, di dalam .NET MAUI terdapat beberapa cara untuk melakukan navigasi, khususnya, kita akan melihat navigasi berbasis shell URI.  Saya berbicara tentang shell di video sebelumnya yang memberikan struktur pada aplikasi kita, tetapi yang hebat dari shell adalah bahwa shell tidak hanya menyertakan layanan untuk melakukan injeksi konstruktor dan injeksi ketergantungan, tetapi juga memungkinkan kita untuk melakukan navigasi berbasis URI.  URI, mirip dengan situs web. Anda pergi ke slash details dan akan masuk ke halaman detail. Anda juga dapat mengirimkan parameter kueri yang dapat berupa tipe data sederhana atau objek yang kompleks.  Hal ini sangat mudah dilakukan dengan .NET MAUI hanya dengan beberapa baris kode. Mari kita beralih ke Visual Studio 2022 dan memulai. Sejauh ini, kami mulai membangun aplikasi Tugas Saya. Di sini, kita memiliki antarmuka pengguna XAML lengkap dan binding data.  Saya dapat menambahkan berbagai item di dalamnya, dan mendapatkan animasi yang bagus. Saya bahkan bisa menggeser untuk menghapus. Sekarang, ini sangat bagus, tapi kami juga ingin bisa mengetuk sebuah item dan melihat detailnya.  Yang akan kita lakukan sekarang adalah menambahkan navigasi ke aplikasi kita.  Mari kita lanjutkan dan lakukan. Hal pertama di sini adalah bahwa ini adalah aplikasi yang sama dengan yang telah kita buat di seluruh seri pemula ini.  Kita memiliki Halaman Utama, kita memiliki Gambar, Entri, Tombol, dan CollectionView di sini.  Kita telah menambahkan SwipeView sehingga kita dapat menambahkan gesekan untuk menghapus.  Kita juga memiliki MainViewModel, yang pada dasarnya adalah kode di balik halaman yang melakukan berbagai properti yang dapat diamati yang dapat kita gunakan untuk melakukan pengikatan data.  Jadi, properti teks yang kita lihat di sana memiliki pengikatan data ke properti teks ini di sini, di Entry.  Mari kita tambahkan halaman lain yang dapat kita navigasikan. Saya akan pergi ke depan dan saya akan mengklik kanan pada MauiApp2 dan mengatakan Tambah, Item Baru.  Ketika saya melakukan itu, saya akan dibawa ke layar ini yang memungkinkan saya untuk memilih item yang berbeda, dan bahkan ada subkategori khusus untuk .NET MAUI.  Di sini kita memiliki halaman dan tampilan serta sumber daya yang berbeda. Saya akan memilih ContentPage XAML untuk .NET MAUI, dan kita akan menyebutnya DetailPage.  Sekarang, halaman ini adalah halaman yang sangat sederhana yang hanya mengatakan "Selamat datang di .NET MAUI."  Itu dia. Sempurna. Tentu saja tidak ada kode di belakangnya, sangat sederhana. Sekarang, hal lain yang akan kita lakukan adalah memasangkannya dengan model tampilan.  Saya akan masuk dan mengatakan Add. Saya akan mengatakan Item Baru di sini, dan di bawah Code, kita akan menambahkan kelas, dan kita akan mengatakan DetailViewModel.  Saya suka menamai model tampilan saya dengan nama yang sangat mirip dengan halaman, jadi DetailPage, DetailViewModel. Saya telah menambahkan kelas baru, jadi kita akan menambahkan beberapa fitur C# 10 yang bagus di sini, seperti file-scoped namespaces. Kita juga akan menjadikannya sebagai kelas parsial publik.  Kita juga akan mewarisi dari ObservableObject. Objek yang dapat diamati tersebut berasal dari toolkit MVVM untuk .NET dan MVVM.  Toolkit komunitas tersebut memberi kita semua yang kita butuhkan untuk mulai melakukan MVVM di sini, dan ini sangat bagus.  Selanjutnya, yang perlu kita lakukan adalah mengaitkan DetailViewModel ini dengan halaman kita.  Di dalam kode DetailPage.xaml di belakang, yang akan kita lakukan adalah di dalam konstruktor, katakanlah DetailViewModel vm, pastikan kita masuk dan membawa namespace tersebut, dan kemudian kita akan mengatur konteks pengikatan di sini ke vm.  Sempurna. Sekarang, seperti yang kita lakukan untuk MainPage dan MainViewModel, kita juga akan mendaftarkannya di program MAUI.  Ini akan mendaftarkan mereka dengan layanan ketergantungan kita sehingga ketika kita menavigasi ke sana, secara otomatis akan mengasosiasikan DetailPage dengan  DetailViewModel.  Sekarang perbedaannya di sini adalah bahwa sebelumnya kita menambahkan singleton, tetapi sebenarnya setiap kali kita menavigasi ke halaman baru, kita ingin membuat halaman baru.  Kita akan mengubahnya menjadi AddTransient. Singleton dapat Anda anggap sebagai global di mana ia membuat satu salinannya dan kemudian akan terus menyimpan memori tersebut, sedangkan transient akan dibuat dan kemudian dihancurkan dan yang baru akan dibuat setiap saat.  Inilah yang kita inginkan untuk terjadi setiap kali kita bernavigasi. Sekarang yang akan kita lakukan adalah mendaftarkan DetailPage dengan sistem perutean.MAUIShell.  Di sini, di dalam AppShell kita, yang sekali lagi jika Anda ingat di sini memiliki ShellContent dari MainPage.  MainPage memiliki rute dari MainPage, yang sangat nyaman.  Saya suka menamai rute saya sama dengan MainPage di sini.  Kita akan masuk dan mendaftarkan halaman ini dalam kode di belakang karena ini bukan bagian dari shell.  Jika saya memiliki tab atau saya harus terbang keluar, itu akan menjadi bagian dari shell, tetapi di sini kita hanya ingin menavigasinya.  Mari kita lanjutkan dan lakukan. Apa yang akan kita lakukan sekarang adalah kita akan mengatakan Routing.RegisterRoute dan ini akan mengambil dua hal.  Pertama adalah rute itu sendiri, jadi saya bisa mengatakan DetailPage. Saya bisa memberikannya DetailPage. Ini bisa menjadi detail. Bisa apa saja yang saya inginkan untuk stringnya.  Saya sering menggunakan nama properti karena di C# jika kita melakukan refactor, maka akan turun ke bawah.  Kemudian kita akan mengasosiasikannya dengan sebuah tipe, jadi kita akan mengatakan typeofDetailPage.  Itu dia. Ini sangat bagus.  Ini memberi saya sebuah string dari DetailPage untuk dinavigasi dan sebuah tipe dari DetailPage untuk dinavigasi juga.  Sempurna. Sekarang mari kita lakukan beberapa navigasi sebelum kita benar-benar mengirimkan data. Inilah yang akan kita lakukan.  Kita akan masuk ke halaman utama, dan ada beberapa cara untuk menambahkan navigasi ketika kita mengklik sebuah item di CollectionView.  Di sini kita dapat mengatakan selected dan kita dapat melihat bahwa ada peristiwa perubahan pilihan dan mode pilihan.  Itu adalah salah satu cara jika kita ingin mendaftarkan sebuah event ke dalam CollectionView, yang merupakan sebuah pilihan.  Sekarang, saya sebenarnya akan mengatakan SelectionMode tidak ada di sini karena kita tidak ingin menggunakan mode seleksi bawaan.  Karena kita hanya melakukan navigasi, kita tidak benar-benar memilih apa pun, saya lebih suka mengetuk item.  Apa yang ingin saya lakukan di sini pada Frame itu sendiri, yang ditempatkan di label ini, adalah saya ingin menambahkan TapGestureRecognizer.  Fitur hebat lainnya adalah ada banyak sekali pengenal gerakan bawaan dan satu yang secara khusus akan kita gunakan adalah ketukan, tetapi ada juga gesek, cubit, geser, jatuhkan, dan seret, tetapi di sini kita akan menggunakan TapGestureRecognizer.  Sekarang, di dalam sini, Anda dapat menetapkan ketukan untuk itu, atau secara khusus perintah, yang sangat bagus, mirip dengan SwipeView kita di atas.  Anda juga dapat mengatakan jumlah ketukan yang diperlukan. Jika Anda memiliki sesuatu yang perlu diketuk 10 kali, Anda bisa melakukannya juga.  Saya akan menyederhanakan hal ini di sini dan saya hanya akan menyalin dan menempelkan perintah dari gesekan untuk menghapus, lalu kita akan mengubahnya.  Kita telah melihat sebelumnya bahwa kita perlu mendapatkan akses dan mengatakan, tolong lakukan beberapa data yang mengikat tumpukan ke MainViewModel kita, dan di sini apa yang akan kita lakukan adalah alih-alih menggunakan DeleteCommand, mari kita lanjutkan dengan menggunakan TapCommand, misalnya.  Di sini yang akan kita lakukan adalah memasukkan parameter berupa string yang diikat ke label di sini.  Sekarang di MainViewModel kita, kita perlu menambahkannya ke perintah tersebut.  Kita memiliki DeleteCommand di sana, mari kita lanjutkan dan tambahkan Tugas asinkronisasi lain, kali ini Tap pass dalam string s. Ini dia.  Sekarang yang akan kita lakukan adalah membuat ICommand yang sekali lagi akan melakukan pembuatan sumber untuk kita sehingga menghasilkan perintah Tap untuk kita.  Sekarang apa yang akan kita lakukan adalah kita akan melanjutkan dan melakukan navigasi.  Di sini saya akan mengatakan tunggu. Sekali lagi, kita melakukan tugas asinkronisasi ini karena ini benar-benar melakukan navigasi, dan ini akan menggunakan navigasi shell, jadi Shell.Current.GoToAsync.  Nah, yang saya sukai dari hal ini adalah saya bisa mengopernya ke DetailPage atau pengenal rute yang saya definisikan di AppShell di sini.  Saya juga bisa mengoper pada string secara langsung, yang juga sangat bagus.  Jika saya ingin, saya bisa kembali ke MainViewModel saya di sini dan mengganti string yang sudah dikodekan dengan nama DetailPage.  Sekarang, ini akan memberi tahu shell untuk membuka DetailPage secara langsung, yang sangat mengagumkan.    Mari kita lanjutkan dan kompilasi ini dan mulai mengetuk item dan kemudian melakukan navigasi ke DetailPage tersebut.  Sekali lagi, ini terserah Anda jika Anda ingin mereferensikan DetailPage di sana atau jika Anda ingin menjadi string atau memiliki beberapa asosiasi, sepenuhnya terserah Anda dan arsitektur aplikasi Anda.  Sekarang mari kita luangkan waktu sejenak untuk mengkompilasi aplikasi dan menerapkannya. Aplikasi kita diluncurkan di sini, dan mari kita lanjutkan dan tambahkan item ke daftar kita.  Sekali lagi, apel, ketika Anda menambahkan, tambahkan itu dan saya akan melakukan tap event, dan bum, begitu saja, kami telah menavigasi ke halaman baru dan kami dapat melihat selamat datang di .Net MAUI.  Sekarang, hal lain yang dapat saya lakukan adalah, tentu saja, memodifikasi XAML itu, jadi jika saya ingin mengatakan ukuran font 25, misalnya, saya menekan "Simpan" dan tentu saja itu akan diperbarui di sana.  Kita juga bisa menambahkan sedikit padding di sini, katakanlah 20. Sekarang kita mendapatkan sedikit padding di sana, sempurna.  Nah, itu cukup bagus karena kita baru saja melakukan beberapa navigasi, tetapi tentu akan lebih bagus lagi jika kita memberikan informasi halaman detail yang sebenarnya, jadi misalnya, item yang saya pilih, dan juga mungkin menambahkan tombol di sini yang mengatakan kembali.  Itu akan sangat bagus. Mari kita lanjutkan dan lakukan selanjutnya.  Inilah yang akan kita lakukan, yaitu kita akan berhenti melakukan debug, kita akan masuk ke model tampilan utama kita lagi, dan sekarang apa yang akan kita  akan kita lakukan adalah kita akan memberikan informasi dan ada dua cara untuk melakukannya.  Cara pertama adalah menambahkannya sebagai properti kueri, dan ini sangat mirip dengan URL, jadi misalnya, saya dapat melanjutkan dan melakukan interpolasi string di sini.  Di sini saya akan melakukan beberapa halaman detail dan kemudian apa yang akan saya lakukan adalah saya akan membuat tanda tanya dan kemudian saya akan membuat teks sama dengan.  Itu akan menjadi pengenal kueri saya, jadi itulah ID di sana. Kemudian saya dapat memasukkan s misalnya, yang merupakan string yang disadap  di sini untuk meneruskannya.  Ini sangat penting karena ini memungkinkan saya untuk mengirim tipe data sederhana seperti string dan bilangan bulat.    Tetapi bagaimana jika Anda memiliki tipe data yang kompleks, seperti orang atau monyet atau mobil atau objek data lainnya?  Hebatnya, Anda tidak hanya dapat mengoper melalui properti kueri di sini, tetapi Anda juga dapat mengoper dalam kamus objek string baru, yang merupakan nilai kunci Anda.  Sebagai contoh, di sini, silakan saja mengisinya, misalnya saya bisa mengatakan, apa halaman detailnya dan menanyakan misalnya, atau saya bisa memasukkan objek baru dari sesuatu, apa pun yang ada di dalam sini, dan Anda bisa meneruskan informasi tersebut.  Jika Anda memiliki tipe data yang kompleks, cara kerjanya sama persis dengan tipe data sederhana, tetapi Anda memiliki kamus ember besar di sini, yang sangat keren.  Saya akan menautkan ke dokumentasi di catatan acara dan bagaimana cara melakukannya juga.  Tapi saya akan langsung saja mengoper properti data sederhana ke string.  Bagaimana kita akan memasukkan string tersebut ke dalam model tampilan detail kita?  Nah, kita dapat menambahkan properti kueri di salah satu dari dua tempat.  Kita dapat melakukannya di halaman detail kita, yang akan baik-baik saja jika kita tidak memiliki MVVM dan model view apapun, tetapi kita juga dapat menambahkan langsung  langsung ke dalam model tampilan kita sendiri.  Selain itu, kita akan menambahkan properti query.  Ini sangat keren dan properti query mengambil sebuah nama, yang akan menjadi nama properti di dalam model view detail yang akan  bind to jadi kita akan mengatakan teks, itulah yang akan kita namakan dan ID properti query teks juga.  Sekali lagi, properti yang dimasukkan di sini akan sama persis dengan properti yang kita daftarkan di sini.  Jika ini adalah ID, misalnya, maka Anda mengubahnya menjadi ID, begitu saja.  Saya suka menamai mereka dengan nama yang sama. Itulah saya. Itu membuat saya lebih mudah mengingatnya. Tapi sekarang kita perlu membuat properti teks ini dan kita  akan membuat satu yang akan kita gunakan untuk melakukan pengikatan data.  Saya hanya akan mengatakan teks string dan saya akan menandainya sebagai properti yang dapat diamati.  Sekali lagi, alasan mengapa pengikatan data ke teks ini akan berfungsi adalah jika kita masuk ke dependensi dan melihat penganalisis untuk komunitas kita  toolkit yang kita bawa akan menghasilkan properti tersebut untuk kita secara otomatis, jadi ada teks dan melakukan semua perubahan untuk kita, yang sangat bagus.  Membuat kode kita sangat sederhana dan itulah mengapa saya sangat menyukai .Net Community Toolkit untuk MVVM.  Sekarang mari kita tambahkan satu hal lagi, mungkin cara untuk menavigasi kembali.  Saya akan melakukan tugas asinkronisasi lain di sini dan mengatakan, kembali.  Itu dia. Sekarang apa yang akan saya lakukan adalah menunggu di Shell.Current.GoToAsync.  Sekarang untuk kembali ke belakang, saya tidak perlu mereferensikan halaman utama karena saya akan mendorong halaman ke stack.  Saya ingin menavigasi ke belakang dan beberapa cara bagaimana Anda menavigasi sistem berkas, saya akan menggunakan tanda titik, dan begitulah cara saya  untuk menavigasi ke belakang.  Anda juga bisa melakukan navigasi tambahan kapan saja dengan menambahkan garis miring ke depan, sehingga akan menavigasi ke belakang dan kemudian ke halaman lain, jadi jika saya memiliki detail atau saya memiliki halaman monyet, misalnya, itu akan menavigasi ke halaman itu, dan Anda dapat melakukannya beberapa kali, misalnya, naik dua halaman dan kemudian juga melewatkan parameter bolak-balik.  Itu sangat bagus. Sekarang kita juga akan menandai ini sebagai ICommand karena itu akan berlanjut dan mengubahnya menjadi perintah untuk saya dan itu akan disebut perintah Go Back, yang akan secara otomatis dibuat dari sumber.  Sekarang yang akan saya lakukan adalah masuk sekali lagi ke halaman detail, dan alih-alih membuat Welcome to MAUI, kita akan melakukan beberapa pengikatan data.  Kita juga perlu melakukan asosiasi dengan tipe data kita, sehingga data tersebut dikompilasi dan diikat. Saya akan membuat sebuah namespace xmlns baru, sebuah model view.  Di sini, saya akan mengetikkan model view dan akan menemukan bahwa model tersebut ada di namespace ini, jadi kita membuat namespace XAML dan saya akan mengatakan tipe data x dan itu akan menjadi model view ke model view detail.  Sekarang yang bisa saya lakukan adalah melakukan pengikatan di sini ke properti teks. Itu dia. Sekarang, mari kita juga membuat tombol di sini dan mengatakan teks, kembali, dan kemudian apa yang akan saya lakukan adalah mendaftarkan perintah ke pengikatan perintah Kembali seperti itu.  Sempurna. Sekarang, mari kita lanjutkan dan lakukan pengikatan data lagi. Untuk semua pengaturan, lihat navigasi kita. Apa yang ingin kita lihat adalah item apa pun yang saya ketuk,  itu harus terus maju dan menavigasi ke halaman itu dan kemudian secara otomatis mengatur properti teks ketika saya menavigasi.  Sekarang aplikasi kita sudah aktif dan berjalan. Saya akan mengatakan apel telah ditambahkan di sana. Saya akan menambahkan jeruk juga. Ketika saya mengetuk salah satunya, tentu saja, kita memiliki apel dan perintah Go Back yang menavigasi ke belakang.  Jeruk kembali ke belakang, apel kembali ke depan. Kita baru saja mengimplementasikan navigasi ke dan dari halaman, navigasi kembali, dan kemampuan untuk mengoper parameter kueri  atau objek penuh menggunakan sistem navigasi bawaan .Net MAUI.  Itu dia, Anda baru saja melakukan beberapa navigasi di aplikasi .Net MAUI Anda. Kita telah mengoper data dari halaman A ke halaman B, dan kita telah menemukan cara untuk menavigasikannya kembali.  Sekarang, apa yang akan kita lakukan selanjutnya dalam Seri Pemula .Net MAUI adalah melihat bagaimana kita dapat mengintegrasikan API platform asli.  Sekarang .NET MAUI memberi Anda semua itu di dalam C-Sharp tetapi juga memberi Anda abstraksi fitur platform umum kami seperti sensor, lokasi geografis, dan konektivitas.  Nantikan video berikutnya dalam Seri Pemula .NET MAUI untuk mempelajari lebih lanjut. |
|  |  |