**INSTALASI VISUAL STUDIO**

VIDEO: <https://www.youtube.com/watch?v=LH4z74QBWU8&list=PLdo4fOcmZ0oUBAdL2NwBpDs32zwGqb9DY&index=3>

|  |  |
| --- | --- |
| **ENGLISH** | **INDONESIA** |
| Today we're continuing our journey, learning how to build native cross-platform desktop and mobile apps with .NET MAUI.  Previously we learned what .NET MAUI is and how to install .NET MAUI as well.  Now what we're going to do in this video is we're going to create our very first project with .NET MAUI and I'm going to walk through all of the important aspects of what is inside of that single project, including a bunch of different cross-platform APIs and frameworks and a bunch of great things to help you be super productive.  Let's head over to my desktop and get started.  When you launch Visual Studio 2022 for the first time, you'll be able to create a brand-new project.  It will be all the projects inside Visual Studio 2022 that you can build based on the workloads that you selected, but we're going to find the .NET MAUI ones by typing in .NET MAUI.  Here we're going to see three different project templates; we have the .NET MAUI app, we have the .NET MAUI Blazor Hybrid application, and a .NET MAUI Class Library.  The class library is nice when you want to share classes and different elements between different .NET MAUI applications.  But let's get started just with the .NET MAUI app.  I'm going to go ahead, hit "Next" and we're going to give it a name.  MauiApp2, that sounds perfect.  I'm going to go ahead and create this and I want to walk through exactly every single bit that is inside of a .NET MAUI application.  Visual Studio has launched.  Now, we can see on the right-hand side, our solution explorer.  Over here we have one project of MauiApp2 because I named it MauiApp2.  What's great here is this gives you all of your dependencies and frameworks in a single project.  That means from the MauiApp2 project, I'm deploying to Android, iOS, Mac, and Windows.  Now, let me walk through some of the other items that are inside this project.  The first thing that we'll see here is the platforms folder.  Now, this is great because this enables us as developers to access platform-specific native APIs.  There is a little bit of scaffolding code in each of them, such as your Android manifest that defines different permissions and app resources, and if your application supports the right to left.  There's also a little bit of startup code like the main activity, and the .NET MAUI team has done a fantastic job of minimizing this boilerplate code as much as humanly possible.  But know that if you need to tweak something on the platform, you can access it right here.  Now, another folder that I love is the resources folder.  Inside this are shared cross-platform resources such as fonts, images, and raw assets.  Now, I love this because we not only get to put all of our fonts and images into a single project here, .NET MAUI will automatically put those into the correct places when it compiles it out for each platform.  Notice that the .NET MAUI bot is an SVG.  Look at it all those paths.  But when you compile the application, it will automatically convert them into PNGs and scale them so they look great on all of your devices.  I'll also note that there are some SVGs for the app icon and foreground as well.  I'll get to those in a little bit.  Now, what I love here is that everything is in this one area, including our app, our shell, and our main page.  Now, before I get to those, let's double-tap on the MauiApp2 because what I want to show you is that there are some amazing cross-platform capabilities built directly into the project system itself.  First and foremost, we're doing multi-targeting.  Here we have Android, iOS, and Mac.  Now, the Windows platform also includes Windows.  Here we'll note that the Samsung Tizen platform is also supported by the Samsung team.  If you want to develop the Tizen devices, you can just uncomment this code and install the Tizen tooling.  Now, as we scroll down, we're going to see a few other properties that are cross-platform properties such as the application title, the identifiers, and even our version codes.  What's great about this is that you can set these in one place and for each platform, they will automatically cascade down so they are automatically set when you compile and deploy your application.  This keeps your version codes, names, and identifiers in sync across all of your applications.  There's another great resource here too, which is this supported OS version, so you can add backward and forward compatibility.  I also want to scroll down a little bit here to this item group.  Now, this is where those resources come in.  I mentioned that there was an app resource and a foreground SVG.    Those are used for our app icon and our splash screen and those are all cross-platform and generated for us automatically.  Here's where our images are coming in.  Automatically, it will bring in just any image that you put in that folder, but you can also include a single SVG, PNG, or JPEG and also update it as well with a base size.  That's great for as SVGs.  Here are also telling it exactly where the fonts are located and also the raw asset.  If you want to, you can put fonts in multiple folders, you could specify different things and automatically this will pick up all the fonts and all the assets for us. That's it.  It's very minimal, which I love.  Now, at the start of any application is the MauiProgram, and this is the scaffolding of the application.  Some are in MauiProgram and the startup code calls and returns a MauiApp.  This is creating a builder.  What's nice about this is this a very similar pattern to ASP.NET Code.  Here it's going to go ahead and create that builder, tell it that it's using this app, and it's going to configure fonts.  There are a lot of other things that you can configure as well, such as activity, lifecycles, services, and the dependency service.  What is in this app?  If we go into the app here, we can see that the app has some app-wide resources in both colors and styles.  Those are located in the resources folder.  Here are the colors which are all defined in our XAML right here.  We can see colors in solid brushes and different accent colors too.  This is great because these are going to be used in our styles, which is the full style set that automatically will style every single control that's built into .NET MAUI.  This means all you need to do is modify a few colors and all of the different controls will update based on what your application needs to look like.  Now, you can hit this little drop-down arrow and you'll see some code behind here.  We usually call this code behind and it's called XAML.CS.  Every XAML.CS is associated with the XAML, and we can see here that the main page of the application is being set to the App Shell.  Well, what's an App Shell? I'm glad you asked.  Well, an App Shell specifically is an opinionated shell of your application.  What's great here is that it enables content templates that are lazily loaded when your application loads up.  Here, it's using a single shell piece of content which is a page, but you can easily add flyout navigation or top and bottom tabs by just adding more items.  Here, I can add a tab and a tab bar, or I can add a flyout item.  I can even add menu files as well.  It's flexible and it also enables your eye-based navigation.  Here, we can see that the route of the app is set to the main page.  That's going to be my main route.  Now, there's this main page and this main page will get inflated with home.  If we tap on the main page, this is XAML.  It's an XML-based markup.  What we have in this application is a scroll view, one of those layouts with a vertical stack layout inside of it.  Now, this is going to go ahead and stack up some images, some labels, and some buttons here to automatically click.  Now, in the code behind as well, we'll see that there is a little bit of code which we can see automatically not only increments the count but will automatically set the text based on the count.  If it's one, it'll say click one time or click multiple times.  Let's run this application now that we're ready.  What I want to show you here is that from the drop-down debug menu, select the target framework that you want to debug on Android, iOS, Mac, and Windows.  Now, Android and Windows can be deployed directly on your Windows device.  Additionally, if you're using Windows 11, you can enable the Windows subsystem for Android to not even have to use an emulator at all.  Use your Windows device as the device that you're debugging on for Android apps.  That's great because both Android and Windows can automatically deploy directly on your Windows 11 device.  If you have Windows 10, don't worry because there's a full Android Device Manager.  Here under tools, we'll see Android and then there are Device Managers as well.  Now, under here when you toggle between the framework of Android, iOS, Mac, or Windows, it will automatically update what you're deploying to.  You'll also see Android emulators, local devices, simulators, and remote devices.  Now, I do want to mention here that for iOS, there are two different ways of deploying the application.  You can either connect remotely to a Mac and deploy and debug through a remote simulator.  Or you can plug an iOS device directly into your Windows machine and use iOS Hot Restart to deploy directly to it if you have an Apple developer account.  For Mac, you need to be on a Mac.  Check out the documentation to deploy there.  Let's go ahead and debug this application right here on Windows.    This is going to go ahead and debug up and compile our application, and then deploy it directly onto my Windows device.  Now, I have it running.  I snapped it right here.  You can see that I simply come in and I am clicking the button and you can see it says, clicked how many times I have it updated.  We have our .NET bar image that was converted from an SVG directly into a PNG saying, Hello world.  Now, there's some other great functionality that's built into Visual Studio.  For example, let's say I have multiple monitors and I don't want the application to overlap on top of it.  I can come into the XAML live preview and pin it here.  What's great is that I can come in and modify the size, and get that window directly inside Visual Studio.  I can still interact with it in the real window that's running, but I can see a preview of it right here.  Now, what that means is that I could go ahead and open up, let's say the main page.  I can come in and say hello to .NET MAUI.  I'm going to hit "Save" and XAML Hot Reload will kick in.  Now, you can see it says, Hello .NET MAUI.    I can also increase that font to 100.  Now, it's huge and I bring over the application.  You can see the application running right there that enables me to be super duper productive.  I'll also mention here that, on the left-hand side, I have this live visual tree.  Now, this is going to show me every single piece of content inside of it.  Here I have my label.  We're going to pin this here.  I have a vertical stack layout, and I have this button, and if I tap on it, it's going to automatically direct me to that button, for example.  That is nice when I'm doing this.  Now, what I also want to show you, go and minimize this over.  If I bring over the application, as I hover, it automatically updates here what element I have selected.  Nice little hints right there.  Now, we've gone ahead and deployed our application to Windows, but how about Android?  Here we go. Going to go over to Android, and now this is going to show me my Android emulators.  If I don't have one, it'll help me create one.  I'm just going to go ahead and debug.  What's great here is, I am debugging my application.  If I want to, I can add breakpoints and I could take advantage of the great capabilities of Visual Studio in debugging and output and watch Windows too.  Now, this is going to start on my Android emulator and we're going to get the same application, but running inside of Android.  Let's give it a few seconds to start up, compile, and deploy.  Here, the application is now running on Android and you can see that it's huge.  Let's go ahead and fix this up.  I'm going to go over that label,  and let's put that back down to 32 and hit ''Save.''  That's going to automatically update.  Of course, I can click here, I go ahead and get a breakpoint and we can see my count as one, so I can go ahead and step through it.  Let's go ahead and remove that.  Awesome. Now, our application is updated and I'm clicking around.  Now, the other thing that's built-in not only is  XAML Hot Reload, but .NET Hot Reload.  Here, let's go ahead and change it to add 10 instead of one every single time.  You can hit ''Save'' or hit this little reload button here, and you can see that I'm hitting hot reload on file save.  Now, when I click on this, it's increasing the counter by 10 every single time.  I can now only update my XAML with hot reload, but also my code behind too.  There you have it. We just started building our very first .NET MAUI application and learn everything that's inside of the project and how to deploy it onto Android or Windows.  Now, let's go ahead and hop over to my Mac where we can see we take the same project and deploy it to both iOS and Mac OS.  Here we go. I've launched the same file new template over on the Mac.  You can do this via the command line or Visual Studio for Mac 2022, and here we can see we have the same application now running on Mac OS and over on our iOS simulator.  You can also plug your iPhone into your Mac and then launch the application directly onto your iOS device.  The same would be true if you wanted to launch the application on an iPad application simulator or on your iPad itself. There you go.  We've not only created our very first project, but we've deployed it to four different operating systems, and we saw the consistent user interface that is being created.  Hopefully, now you have a full understanding of what is inside of a.NET MAUI application when you get started.  What we're going to do in the next video is, we are going to learn how to build out a more complex user interface and handle user interaction.  Stay tuned to the .NET MAUI Beginner Series. | Hari ini kita akan melanjutkan perjalanan kita, mempelajari cara membuat aplikasi desktop dan mobile lintas platform dengan .NET MAUI.  Sebelumnya kita telah mempelajari apa itu .NET MAUI dan bagaimana cara menginstal .NET MAUI.  Sekarang apa yang akan kita lakukan dalam video ini adalah kita akan membuat proyek pertama kita dengan .NET MAUI dan saya akan membahas semua aspek penting dari apa yang ada di dalam satu proyek tersebut, termasuk banyak API dan kerangka kerja lintas platform yang berbeda dan banyak hal hebat untuk membantu Anda menjadi sangat produktif.  Mari kita buka desktop saya dan mulai.  Ketika Anda meluncurkan Visual Studio 2022 untuk pertama kalinya, Anda akan dapat membuat proyek baru.  Ini akan menjadi semua proyek di dalam Visual Studio 2022 yang dapat Anda buat berdasarkan beban kerja yang Anda pilih, tetapi kami akan menemukan proyek .NET MAUI dengan mengetikkan .NET MAUI.  Di sini kita akan melihat tiga templat proyek yang berbeda; kita memiliki aplikasi .NET MAUI, kita memiliki aplikasi .NET MAUI Blazor Hybrid, dan Perpustakaan Kelas .NET MAUI.  Perpustakaan kelas sangat bagus ketika Anda ingin berbagi kelas dan elemen yang berbeda antara aplikasi .NET MAUI yang berbeda.  Tapi mari kita mulai dengan aplikasi .NET MAUI.  Saya akan melanjutkan, tekan "Next" dan kita akan memberinya nama.  MauiApp2, kedengarannya sempurna.  Saya akan melanjutkan dan membuat ini dan saya ingin membahas setiap bagian yang ada di dalam aplikasi .NET MAUI.  Visual Studio telah diluncurkan.  Sekarang, kita dapat melihat di sisi kanan, penjelajah solusi kita.  Di sini kita memiliki satu proyek MauiApp2 karena saya menamainya MauiApp2.  Yang hebat di sini adalah ini memberi Anda semua dependensi dan kerangka kerja Anda dalam satu proyek.  Itu berarti dari proyek MauiApp2, saya men-deploy ke Android, iOS, Mac, dan Windows.  Sekarang, mari saya bahas beberapa item lain yang ada di dalam proyek ini.  Hal pertama yang akan kita lihat di sini adalah folder platform.  Ini sangat bagus karena ini memungkinkan kita sebagai pengembang untuk mengakses API asli khusus platform.  Ada sedikit kode perancah di masing-masingnya, seperti manifes Android Anda yang mendefinisikan izin dan sumber daya aplikasi yang berbeda, dan jika aplikasi Anda mendukung kanan ke kiri.  Ada juga sedikit kode startup seperti aktivitas utama, dan tim .NET MAUI telah melakukan pekerjaan yang luar biasa dalam meminimalkan kode boilerplate ini sebanyak mungkin.  Tetapi ketahuilah bahwa jika Anda perlu mengubah sesuatu pada platform, Anda dapat mengaksesnya di sini.  Sekarang, folder lain yang saya sukai adalah folder sumber daya.  Di dalamnya terdapat sumber daya lintas platform yang digunakan bersama seperti font, gambar, dan aset mentah.  Sekarang, saya suka ini karena kita tidak hanya bisa meletakkan semua font dan gambar ke dalam satu proyek di sini, .NET MAUI akan secara otomatis meletakkannya di tempat yang benar saat mengompilasinya untuk setiap platform.  Perhatikan bahwa bot .NET MAUI adalah sebuah SVG.  Lihatlah semua jalur itu.  Tetapi ketika Anda mengkompilasi aplikasi, secara otomatis akan mengonversinya menjadi PNG dan menskalakannya sehingga terlihat bagus di semua perangkat Anda.  Saya juga akan mencatat bahwa ada beberapa SVG untuk ikon aplikasi dan latar depan juga.  Saya akan membahasnya sebentar lagi.  Sekarang, yang saya sukai di sini adalah semuanya ada di satu area ini, termasuk aplikasi, shell, dan halaman utama.  Sekarang, sebelum saya membahasnya, mari kita ketuk dua kali pada MauiApp2 karena yang ingin saya tunjukkan kepada Anda adalah bahwa ada beberapa kemampuan lintas platform yang luar biasa yang dibangun langsung ke dalam sistem proyek itu sendiri.  Pertama dan terutama, kami melakukan penargetan multi-penargetan.  Di sini kami memiliki Android, iOS, dan Mac.  Sekarang, platform Windows juga termasuk Windows.  Di sini kita akan mencatat bahwa platform Samsung Tizen juga didukung oleh tim Samsung.  Jika Anda ingin mengembangkan perangkat Tizen, Anda bisa menghapus kode ini dan menginstal perkakas Tizen.  Sekarang, saat kita menggulir ke bawah, kita akan melihat beberapa properti lain yang merupakan properti lintas platform seperti judul aplikasi, pengenal, dan bahkan kode versi.  Hal yang hebat dari hal ini adalah Anda bisa mengaturnya di satu tempat dan untuk setiap platform, properti-properti tersebut akan secara otomatis turun ke bawah sehingga secara otomatis diatur ketika Anda mengkompilasi dan menerapkan aplikasi Anda.  Hal ini membuat kode versi, nama, dan pengenal Anda tetap sinkron di semua aplikasi Anda.  Ada sumber daya hebat lainnya di sini juga, yaitu versi OS yang didukung, sehingga Anda bisa menambahkan kompatibilitas mundur dan maju.  Saya juga ingin menggulir sedikit ke bawah di sini ke grup item ini.  Nah, di sinilah sumber daya itu masuk.  Saya telah menyebutkan bahwa ada sumber daya aplikasi dan SVG latar depan.    Keduanya digunakan untuk ikon aplikasi dan layar pembuka dan semuanya lintas platform dan dibuat untuk kita secara otomatis.  Di sinilah gambar-gambar kita masuk.  Secara otomatis, ini akan membawa sembarang gambar yang Anda masukkan ke dalam folder itu, tetapi Anda juga bisa menyertakan satu SVG, PNG, atau JPEG dan juga memperbaruinya dengan ukuran dasar.  Ini sangat bagus untuk SVG.  Di sini juga memberi tahu di mana letak font dan juga aset mentahnya.  Jika Anda ingin, Anda dapat meletakkan font di beberapa folder, Anda dapat menentukan hal-hal yang berbeda dan secara otomatis ini akan mengambil semua font dan semua aset untuk kita. Itu saja.  Ini sangat minimal, dan saya suka.  Sekarang, di awal aplikasi apa pun adalah MauiProgram, dan ini adalah perancah aplikasi.  Beberapa ada di MauiProgram dan kode startup memanggil dan mengembalikan MauiApp.  Ini adalah membuat sebuah pembangun.  Yang menyenangkan dari hal ini adalah pola yang sangat mirip dengan ASP.NET Code.  Di sini ia akan melanjutkan dan membuat pembangun itu, memberi tahu bahwa ia menggunakan aplikasi ini, dan akan mengkonfigurasi font.  Ada banyak hal lain yang bisa Anda konfigurasikan juga, seperti aktivitas, siklus hidup, layanan, dan layanan ketergantungan.  Apa saja yang ada di dalam aplikasi ini?  Jika kita masuk ke dalam aplikasi di sini, kita bisa melihat bahwa aplikasi ini memiliki beberapa sumber daya di seluruh aplikasi dalam warna dan gaya.  Semua itu berada di folder sumber daya.  Berikut adalah warna-warna yang semuanya didefinisikan dalam XAML kita di sini.  Kita dapat melihat warna dalam kuas solid dan warna aksen yang berbeda juga.  Ini sangat bagus karena ini akan digunakan dalam gaya kita, yang merupakan kumpulan gaya lengkap yang secara otomatis akan menata setiap kontrol yang ada di .NET MAUI.  Ini berarti Anda hanya perlu memodifikasi beberapa warna dan semua kontrol yang berbeda akan diperbarui berdasarkan tampilan aplikasi Anda.  Sekarang, Anda dapat menekan panah drop-down kecil ini dan Anda akan melihat beberapa kode di belakangnya.  Kita biasanya menyebut kode ini sebagai kode di belakang dan disebut XAML.CS.  Setiap XAML.CS berhubungan dengan XAML, dan kita dapat melihat di sini bahwa halaman utama aplikasi diatur ke App Shell.  Nah, apa itu App Shell? Saya senang Anda bertanya.  Secara khusus, App Shell adalah cangkang aplikasi Anda.  Yang hebat di sini adalah memungkinkan templat konten yang dimuat dengan malas saat aplikasi Anda dimuat.  Di sini, ini menggunakan satu cangkang konten yang berupa halaman, tetapi Anda dapat dengan mudah menambahkan navigasi flyout atau tab atas dan bawah hanya dengan menambahkan lebih banyak item.  Di sini, saya bisa menambahkan tab dan bilah tab, atau saya bisa menambahkan item flyout.  Saya bahkan bisa menambahkan file menu juga.  Sangat fleksibel dan juga memungkinkan navigasi berbasis mata Anda.  Di sini, kita dapat melihat bahwa rute aplikasi diatur ke halaman utama.  Itu akan menjadi rute utama saya.  Sekarang, ada halaman utama ini dan halaman utama ini akan digelembungkan dengan home.  Jika kita mengetuk halaman utama, ini adalah XAML.  Ini adalah markup berbasis XML.  Apa yang kita miliki dalam aplikasi ini adalah tampilan gulir, salah satu tata letak dengan tata letak tumpukan vertikal di dalamnya.  Sekarang, ini akan menumpuk beberapa gambar, beberapa label, dan beberapa tombol di sini untuk diklik secara otomatis.  Sekarang, di dalam kode di belakangnya juga, kita akan melihat bahwa ada sedikit kode yang bisa kita lihat secara otomatis tidak hanya menambah jumlah tetapi juga secara otomatis akan mengatur teks berdasarkan jumlah.  Jika satu, maka akan muncul tulisan klik satu kali atau klik beberapa kali.  Mari kita jalankan aplikasi ini sekarang setelah kita siap.  Yang ingin saya tunjukkan di sini adalah bahwa dari menu drop-down debug, pilih target framework yang ingin Anda debug di Android, iOS, Mac, dan Windows.  Sekarang, Android dan Windows dapat digunakan langsung pada perangkat Windows Anda.  Selain itu, jika Anda menggunakan Windows 11, Anda bisa mengaktifkan subsistem Windows untuk Android sehingga tidak perlu menggunakan emulator sama sekali.  Gunakan perangkat Windows Anda sebagai perangkat yang Anda debug untuk aplikasi Android.  Ini sangat bagus karena Android dan Windows dapat secara otomatis digunakan langsung pada perangkat Windows 11 Anda.  Jika Anda menggunakan Windows 10, jangan khawatir karena ada Android Device Manager yang lengkap.  Di sini, di bawah alat, kita akan melihat Android dan kemudian ada Pengelola Perangkat juga.  Nah, di bawah sini ketika Anda beralih antara kerangka kerja Android, iOS, Mac, atau Windows, maka secara otomatis akan memperbarui apa yang Anda terapkan.  Anda juga akan melihat emulator Android, perangkat lokal, simulator, dan perangkat jarak jauh.  Sekarang, saya ingin menyebutkan di sini bahwa untuk iOS, ada dua cara yang berbeda untuk menerapkan aplikasi.  Anda bisa menghubungkan dari jarak jauh ke Mac dan menerapkan serta men-debug melalui simulator jarak jauh.  Atau Anda bisa mencolokkan perangkat iOS langsung ke mesin Windows Anda dan menggunakan iOS Hot Restart untuk menerapkannya langsung jika Anda memiliki akun pengembang Apple.  Untuk Mac, Anda harus menggunakan Mac.  Lihat dokumentasi untuk menerapkan di sana.  Mari kita lanjutkan dan debug aplikasi ini di sini, di Windows.    Ini akan men-debug dan mengkompilasi aplikasi kita, lalu menerapkannya langsung ke perangkat Windows saya.  Sekarang, saya sudah menjalankannya.  Saya memotretnya di sini.  Anda dapat melihat bahwa saya hanya masuk dan saya mengklik tombol dan Anda dapat melihat bahwa ada tulisan, diklik berapa kali saya memperbaruinya.  Kami memiliki gambar bar .NET kami yang telah dikonversi dari SVG langsung menjadi PNG yang mengatakan, Halo dunia.  Sekarang, ada beberapa fungsionalitas hebat lainnya yang ada di dalam Visual Studio.  Sebagai contoh, katakanlah saya memiliki beberapa monitor dan saya tidak ingin aplikasi tumpang tindih di atasnya.  Saya bisa masuk ke pratinjau langsung XAML dan menyematkannya di sini.  Hebatnya lagi, saya bisa masuk dan memodifikasi ukurannya, dan mendapatkan jendela itu langsung di dalam Visual Studio.  Saya masih bisa berinteraksi dengannya di jendela nyata yang sedang berjalan, tetapi saya bisa melihat pratinjaunya di sini.  Sekarang, artinya saya bisa langsung membuka, katakanlah halaman utama.  Saya bisa masuk dan menyapa .NET MAUI.  Saya akan menekan "Simpan" dan XAML Hot Reload akan dimulai.  Sekarang, Anda dapat melihat tulisan, Hello .NET MAUI.    Saya juga dapat meningkatkan font tersebut menjadi 100.  Sekarang, tampilannya menjadi besar dan saya membuka aplikasinya.  Anda dapat melihat aplikasi berjalan di sana yang memungkinkan saya untuk menjadi sangat produktif.  Saya juga akan menyebutkan di sini bahwa, di sisi kiri, saya memiliki pohon visual yang hidup.  Sekarang, ini akan menunjukkan kepada saya setiap bagian konten di dalamnya.  Di sini saya memiliki label saya.  Kita akan menyematkan ini di sini.  Saya memiliki tata letak tumpukan vertikal, dan saya memiliki tombol ini, dan jika saya mengetuknya, ini akan secara otomatis mengarahkan saya ke tombol itu, misalnya.  Itu bagus ketika saya melakukan ini.  Sekarang, yang juga ingin saya tunjukkan kepada Anda, buka dan minimalkan ini.  Jika saya membawa aplikasi, saat saya mengarahkan kursor, secara otomatis memperbarui di sini elemen apa yang telah saya pilih.  Petunjuk kecil yang bagus di sana.  Sekarang, kita telah melanjutkan dan menerapkan aplikasi kita ke Windows, tetapi bagaimana dengan Android?  Ini dia. Kita akan beralih ke Android, dan sekarang ini akan menunjukkan emulator Android saya.  Jika saya tidak memilikinya, ini akan membantu saya membuatnya.  Saya hanya akan melanjutkan dan melakukan debug.  Yang hebat di sini adalah, saya melakukan debug pada aplikasi saya.  Jika saya mau, saya dapat menambahkan breakpoint dan saya dapat memanfaatkan kemampuan hebat dari Visual Studio dalam debugging dan output dan melihat Windows juga.  Sekarang, ini akan dimulai pada emulator Android saya dan kita akan mendapatkan aplikasi yang sama, tetapi berjalan di dalam Android.  Mari kita beri waktu beberapa detik untuk memulai, mengkompilasi, dan menerapkannya.  Di sini, aplikasi sekarang berjalan di Android dan Anda dapat melihat bahwa aplikasi ini sangat besar.  Mari kita lanjutkan dan perbaiki ini.  Saya akan menghapus label itu,  dan mari kita turunkan kembali ke 32 dan tekan "Simpan.  Itu akan diperbarui secara otomatis.  Tentu saja, saya dapat mengklik di sini, saya lanjutkan dan mendapatkan breakpoint dan kita dapat melihat hitungan saya sebagai satu, jadi saya dapat melanjutkan dan melewatinya.  Mari kita lanjutkan dan hapus itu.  Bagus sekali. Sekarang, aplikasi kita sudah diperbarui dan saya tinggal klik saja.  Sekarang, hal lain yang ada di dalamnya tidak hanya  XAML Hot Reload, tetapi juga .NET Hot Reload.  Di sini, mari kita lanjutkan dan ubah untuk menambahkan 10, bukannya satu setiap kali.  Anda bisa menekan ''Simpan'' atau menekan tombol muat ulang kecil ini di sini, dan Anda bisa melihat bahwa saya menekan muat ulang panas pada penyimpanan file.  Sekarang, ketika saya mengklik ini, penghitungnya akan bertambah 10 setiap saat.  Sekarang saya tidak hanya dapat memperbarui XAML saya dengan hot reload, tetapi juga kode di belakangnya.  Itu dia. Kita baru saja mulai membangun aplikasi .NET MAUI pertama kita dan mempelajari semua yang ada di dalam proyek dan cara menerapkannya ke Android atau Windows.  Sekarang, mari kita lanjutkan dan beralih ke Mac saya di mana kita dapat melihat kita mengambil proyek yang sama dan menerapkannya ke iOS dan Mac OS.  Ini dia. Saya telah meluncurkan file yang sama dengan template baru di Mac.  Anda bisa melakukan ini melalui baris perintah atau Visual Studio untuk Mac 2022, dan di sini kita bisa melihat bahwa aplikasi yang sama sekarang berjalan di Mac OS dan di simulator iOS kita.  Anda juga bisa mencolokkan iPhone Anda ke Mac dan kemudian meluncurkan aplikasi langsung ke perangkat iOS Anda.  Hal yang sama juga berlaku jika Anda ingin meluncurkan aplikasi di simulator aplikasi iPad atau di iPad Anda sendiri. Itu dia.  Kami tidak hanya membuat proyek pertama kami, tetapi kami juga telah menerapkannya pada empat sistem operasi yang berbeda, dan kami melihat antarmuka pengguna yang konsisten yang dibuat.  Mudah-mudahan, sekarang Anda memiliki pemahaman penuh tentang apa yang ada di dalam aplikasi a.NET MAUI ketika Anda memulai.  Apa yang akan kita lakukan di video berikutnya adalah, kita akan belajar bagaimana membuat antarmuka pengguna yang lebih kompleks dan menangani interaksi pengguna.  Nantikan terus Seri Pemula .NET MAUI. |
|  |  |