IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Menggunakan Blazor Hybrid pada WPF dan Windows Forms

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

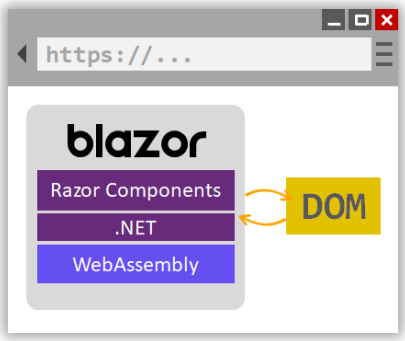
Abstrak

Blazor Hybrid adalah pendekatan inovatif yang menggabungkan framework Blazor dengan .NET MAUI (Multi-platform App UI). Yang memungkinkan kita sebagai *developer* untuk membangun aplikasi *cross platform* menggunakan teknologi web yang sudah dikenal. Dalam aplikasi Blazor Hybrid, komponen Razor berjalan secara native di perangkat dan direnderer ke kontrol Web View yang tertanam melalui local interop. Sehingga komponen ini tidak berjalan di browser dan tidak melibatkan WebAssembly.

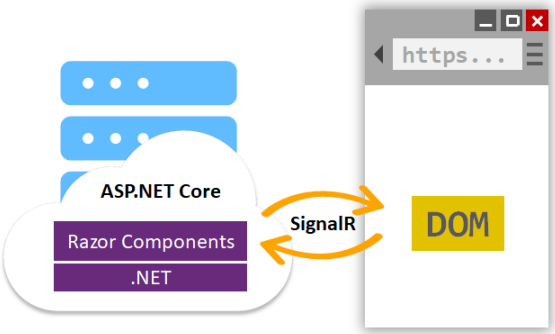
Pendahuluan

Blazor adalah Web Framework yang bersifat Open Source dimana aplikasi Web yang bersifat client-side interactive dapat dikembangkan dengan menggunakan .Net (C#) dan HTML. Pada saat ini C# biasa digunakan untuk melakukan proses back-end dari aplikasi web. Dengan menggunakan fitur baru dari ASP.NET Core yaitu Blazor, kita dapat membangun interactive WEB dengan menggunakan C# dan .NET .

Code .Net berjalan pada Web Assembly, yang artinya kita dapat menjalankan "NET" didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.



Dengan menggunakan Blazor kita dapat memilih apakah menggunakan WebAssemby (Client Side), seperti yang telah dijelaskan di atas atau Blazor yang dijalankan diatas server. Dengan menggunakan cara kedua kita memerlukan SignalR untuk menghubungkan antara client (browser) dan server app. Sebagai contoh jika user melakukan proses klik button pada browser, maka data akan dikirmkan ke Server menggunakan SignalR dan hasilnya akan dikembalikan ke client dengan mengupdate DOM pada client.



Aplikasi Blazor menggunakan runtime .NET yang didasarkan pada Web Assembly atau disingkat WASM. WASM didukung secara native oleh mayoritas browser.

Blazor Hybrid adalah pendekatan inovatif yang menggabungkan framework Blazor dengan .NET MAUI (Multi-platform App UI). Yang memungkinkan kita sebagai *developer* untuk membangun aplikasi *cross platform* menggunakan teknologi web yang sudah dikenal. Dalam aplikasi Blazor Hybrid, komponen Razor berjalan secara native di perangkat dan direnderer ke kontrol Web View yang tertanam melalui local interop. Sehingga komponen ini tidak berjalan di browser dan tidak melibatkan WebAssembly.

Disarankan untuk membaca dan menyeselesaikan latihan pada artikel sebelumnya pada tautan berikut : http://junindar.blogspot.com/2020/02/pengenalan-blazor.html , https://junindar.blogspot.com/2024/12/dialog-component-pada-blazor-hybrid.html , https://junindar.blogspot.com/2024/12/razor-class-library-pada-blazor-hybrid.html dan https://junindar.blogspot.com/2025/02/authentication-and-authorization-pada.html

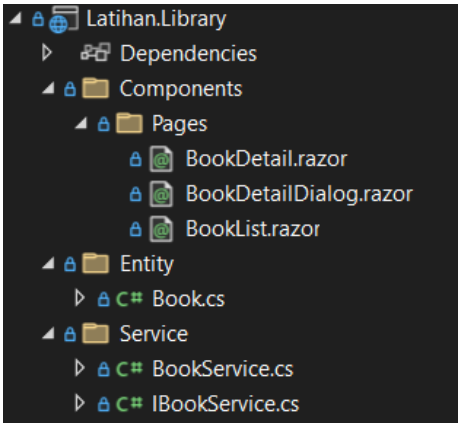
Untuk menggunakan Blazor Hybrid pada WPF dan Windows Forms kita menggunakan Razor Class Library. Pada Bab sebelumnya telah kita bahas bagaimana menggunakan RCL pada Blazor Web maupun MAUI.

Pada pembahasan sebelumnya kita tidak memindahkan seluruh komponen seperti "Home.razor", "Routes.razor" maupun Layout. Sehingga jika kita ingin menggunakan RCL ini pada WPF project, kita perlu meng-copy komponen-komponen tersebut kedalam WPF project. Tapi hal ini akan menjadi masalah jika terdapat banyak code dan digunakan dibanyak project seperti .NET MAUI dan WPF. Sehingga jika terjadi perubahan harus dilakukan dimasing-masing project.

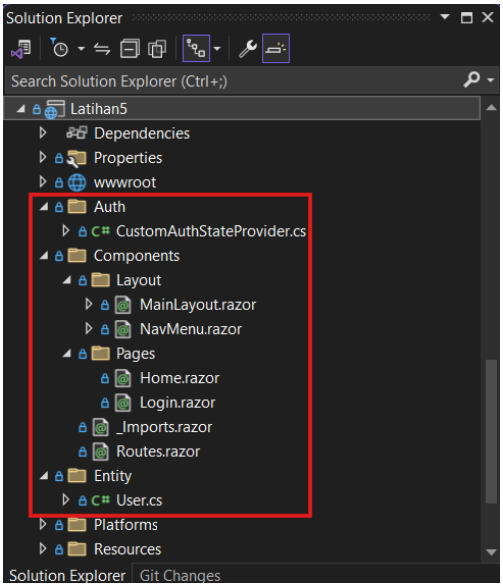
Pada pembahasan ini kita akan membuat RCL yang dapat digunakan dibanyak platform atau project Dimana jika terjadi perubahan code kita tidak perlu untuk mengganti code pada setiap project.

Ikuti Langkah-langkah dibawah ini.

- Buat sebuah Razor Class Library project dengan nama "Latihan.RCL", selanjutnya pada copy semua file dalam Razor Class Library project yang kita buat pada pembahasan sebelumnya kedalam project baru ini.



- Setelah itu buka project "Latihan5" (https://github.com/junindar/ilkom/tree/master/BlazorHybrid/BlazorHybrid/Latihan5), lalu pindahkan seluruh file pada folder "Auth", "Components" dan "Entity" kedalam RCL project yang baru kita buat diatas.



- Buka file "_Imports.razor" dan ganti namespace menjadi Latihan.RCL.

```
@using System.Net.Http
@using System.Net.Http.Json
@using Microsoft.AspNetCore.Components.Forms
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.AspNetCore.Components.Web.Virtualization
@using Microsoft.AspNetCore.Components.Authorization
@using Microsoft.JSInterop
@using Latihan.RCL
@using Latihan.RCL.Components
@using Latihan.RCL.Auth
@using Latihan.RCL.Components.Pages
```

Dan tambahkan "Microsoft.AspNetCore.Components.Authorization" library pada Nuget Package.

- Lalu buka "Routes.razor", komponen ini digunakan untuk mendifinisikan "Router". Dapat kita lihat bahwa properti AppAssembly diatur ke Assembly dari kelas MauiProgram. Namun sekarang, kita berada di Latihan.RCL dimana tidak terdapat class MauiProgram. Sehingga kita perlu mengganti nilai tersebut menjadi Assembly dari class library, karena class library sekarang berisi semua komponen, kita bisa menganti dengan nama salah satu komponen, misalnya, "BookList" komponen untuk menetapkan Assembly dari class library ke Router pada properti Assembly.

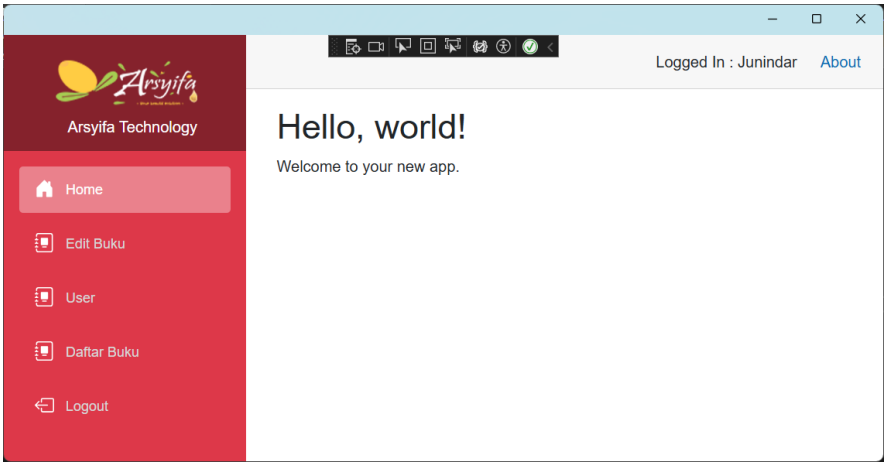
```
<Router AppAssembly="@typeof(BookList).Assembly">
    <Found Context="routeData">
        <AuthorizeRouteView  RouteData="@routeData" DefaultLayout="@typeof(Layout.MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
</Router>
```

- Setelah selesai dengan langkah-langkah diatas, kita akan mencoba untuk menggunakan RCL tersebut kedalam project Blazor Hybrid (Latihan5) terlebih dahulu. Klik kanan pada project BlazorLogin lalu Add > Project Reference dan pilih "Latihan.RCL". Buka File MainPage.xaml dimana di dalamnya terdapat BlazorWebView. Disana kita dapat melihat bahwa elemen RootComponent memuat ComponentType, yang sebenarnya mengambil dari komponen Routes. Dimana secara default menggunakan prefix lokal yang mengarah ke namespace "BlazorLogin".

  Namun sekarang, komponen Routes berada di Latihan.RCL. Jadi kita perlu menghapus namespace dengan prefix lokal tersebut. Dilanjutkan dengan melakukan mapping namespace baru dan gunakan prefix lib untuk library. Kita gunakan namespace "Latihan.RCL.Components " dan map ke prefix lib. Seperti pada sintaks dibawah ini.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:lib="clr-namespace:Latihan.RCL.Components;assembly=Latihan.RCL"
             x:Class="MauiAppWithRCL2.MainPage"
             BackgroundColor="{DynamicResource PageBackgroundColor}">

    <BlazorWebView x:Name="blazorWebView" HostPage="wwwroot/index.html">
        <BlazorWebView.RootComponents>
            <RootComponent Selector="#app" ComponentType="{x:Type lib:Routes}" />
        </BlazorWebView.RootComponents>
    </BlazorWebView>

</ContentPage>
```

Jalankan program dan pastikan mendapatkan hasil seperti dibawah.

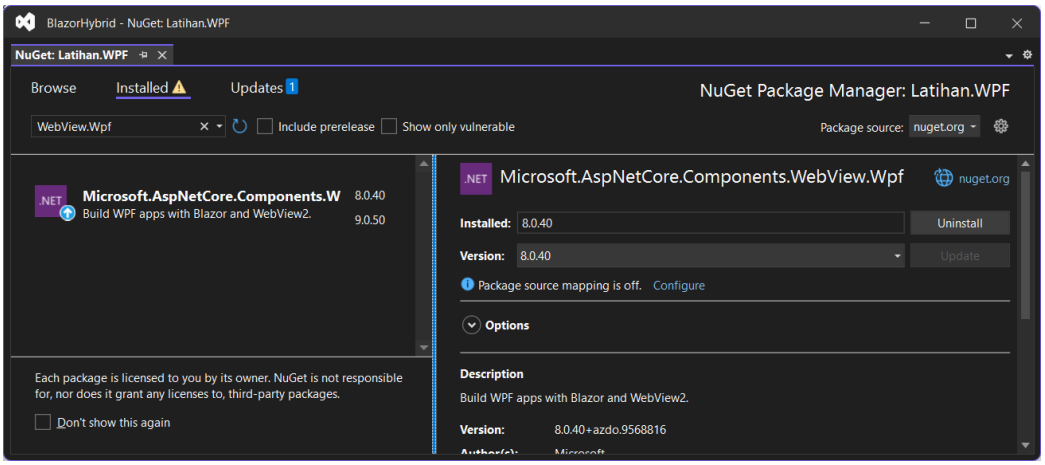


Windows Presentation Foundation (WPF)

WPF (Windows Presentation Foundation) adalah sebuah framework dari Microsoft yang digunakan untuk membangun aplikasi desktop berbasis Windows dengan tampilan yang kaya (rich UI). WPF merupakan bagian dari .NET Framework dan .NET Core/ .NET (terbaru), yang memungkinkan pengembang untuk membuat antarmuka pengguna (UI) modern.

Selanjutnya kita akan mencoba untuk menggunakan RCL kedalam WPF project, ikuti langkah-langkah dibawah ini.

- Tambahkan sebuah WPF project lalu buka CS project file dengan cara double click project yang baru saja kita buat. Setelah terbuka ganti nilai dari "Sdk" pada node Project menjadi "Microsoft.NET.Sdk.Razor". Klik kanan pada project lalu Add > Project Reference dan pilih "Latihan.RCL". Lalu copy folder "wwwroot" pada project BlazorLogin kedalam WPF project.
- Untuk menggunakan RCL pada WPF kita perlu menggunakan BlazorWebView untuk WPF atau "Microsoft.AspNetCore.Components.WebView". Oleh karena itu kita perlu menambahkan library tersebut kedalam project dengan menggunakan NuGet seperti pada gambar dibawah ini.



- Buka file MainWindow.xaml lalu tambahkan sintaks seperti pada gambar dibawah ini.

```
<Window x:Class="Latihan.WPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:lib="clr-namespace:Latihan.RCL.Components;assembly=Latihan.RCL"
        xmlns:wpf="http://schemas.microsoft.com/winfx/2006/xaml/presentation/blazor"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <wpf:BlazorWebView x:Name="blazorWebView" HostPage="wwwroot/index.html"
                           Services="{DynamicResource ServiceProvider}">
            <wpf:BlazorWebView.RootComponents>
                <wpf:RootComponent Selector="#app" ComponentType="{x:Type lib:Routes}" />
            </wpf:BlazorWebView.RootComponents>
        </wpf:BlazorWebView>
    </Grid>
</Window>
```

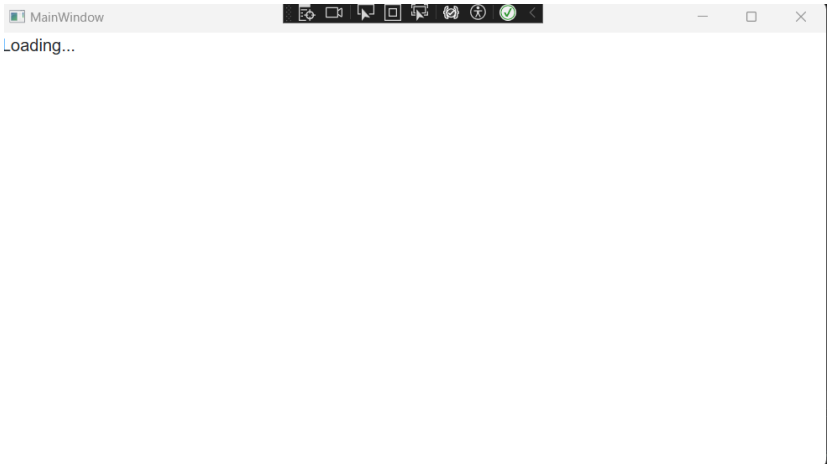
- Setelah selesai dengan langkah diatas, kita lanjutkan dengan bagaimana cara mengatur depedency injection pada WPF project. Buka file App.xaml dan pada Application Element kita tentukan event handler untuk Startup event.

```
<Application x:Class="Latihan.WPF.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Latihan.WPF"
             StartupUri="MainWindow.xaml"
             Startup="Application_OnStartup">
    <Application.Resources>

    </Application.Resources>
</Application>
```

Selanjutnya bukan App.xaml.cs dan ketikkan seperti pada sintaks dibawah ini.

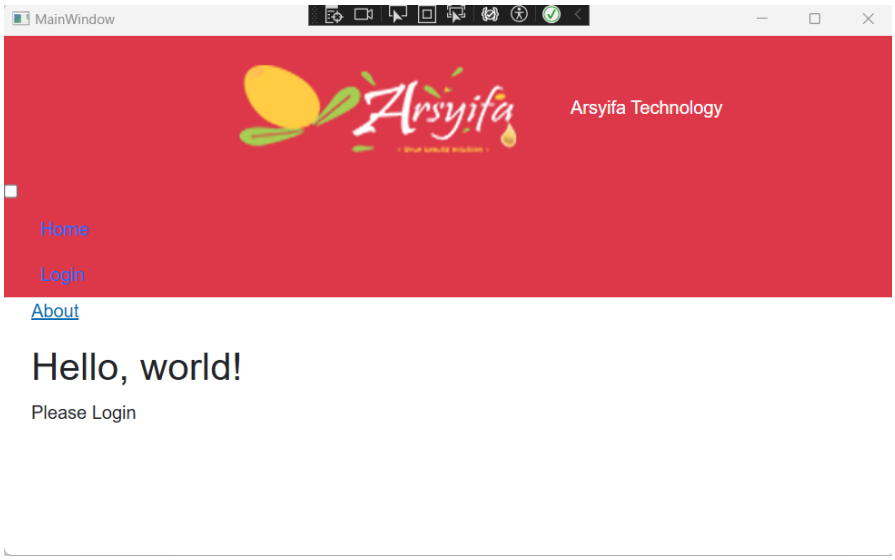
```
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    3 references | 0 changes | 0 authors, 0 changes
    public partial class App : Application
    {
        1 reference | 0 changes | 0 authors, 0 changes
        private void Application_OnStartup(object sender, StartupEventArgs e)
        {
            var services = new ServiceCollection();
            services.AddWpfBlazorWebView();
#if DEBUG
            services.AddBlazorWebViewDeveloperTools();
#endif


            services.AddAuthorizationCore();

            services.AddScoped<AuthenticationStateProvider, CustomAuthStateProvider>();
            services.AddTransient<IBookService, BookService>();
            var serviceProvider = services.BuildServiceProvider();
            this.Resources.Add("ServiceProvider", serviceProvider);


        }
    }
}
```

Dan coba jalankan program.



Jika mendapatkan hasil seperti gambar diatas, kita perlu melakukan perubahan sintaks pada file “index.html” didalam folder wwwroot, lalu hapus sintaks autostart=”false” dan jalankan program.

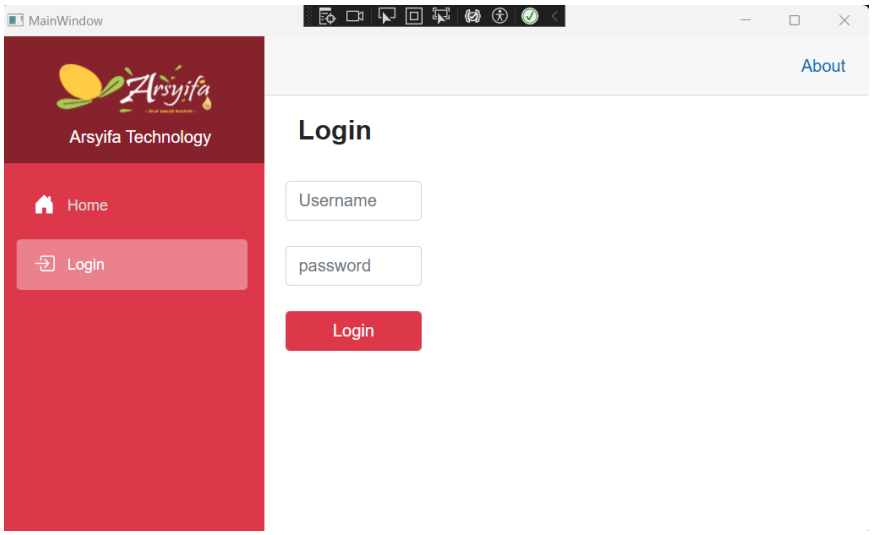
```
<script src="_framework/blazor.webview.js" autostart="false"></script>
```



Program sudah berjalan tanpa error, tapi antara muka atau design dari pada form masih belum sesuai dengan latihan-latihan sebelumnya. Untuk mengatasi masalah tersebut, masih dalam file yang sama “index.html”. Pastikan href telah menggunakan nama project dari WPF yang baru kita buat.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
    <title>Latihan WPF</title>
    <base href="/" />
    <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
    <link rel="stylesheet" href="css/app.css" />
    <link rel="stylesheet" href="Latihan.WPF.styles.css" />
    <link rel="icon" type="image/png" href="favicon.png" />
</head>
```

Dan jalankan program pastikan mendapatkan hasil seperti dibawah



Windows Form

Windows Forms (WinForms) Project di Visual Studio adalah jenis project yang digunakan untuk membuat aplikasi desktop berbasis GUI (*Graphical User Interface*) menggunakan teknologi Windows Forms. WinForms adalah salah satu framework UI di .NET yang memungkinkan pengembang untuk membangun aplikasi dengan antarmuka pengguna berbasis jendela, tombol, kotak teks, dan elemen UI lainnya.

Fitur Windows Forms Project:

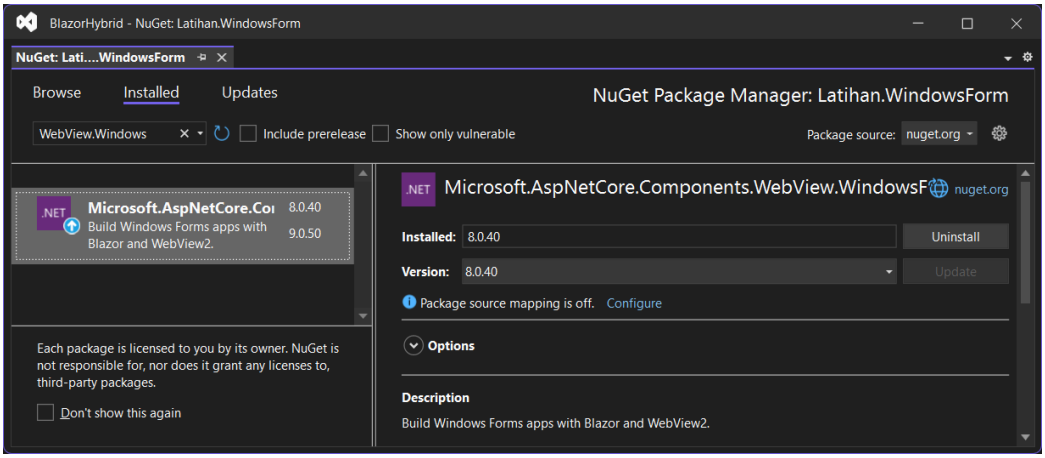
Desainer Visual – Memungkinkan pembuatan UI secara drag-and-drop menggunakan Visual Studio.

Kontrol Bawaan – Seperti tombol, label, textbox, datagridviev dan lain-lain.

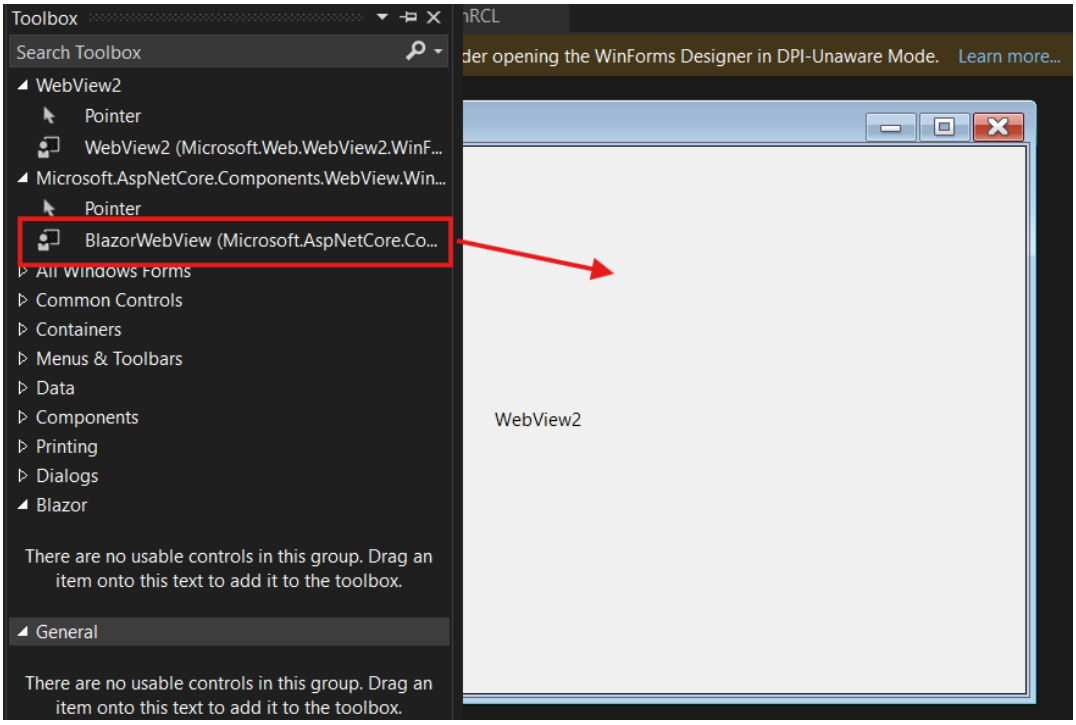
Event-Driven Programming – Menggunakan event seperti Click, Load, dan lainnya untuk menangani interaksi pengguna.

Selanjutnya kita akan mencoba untuk menggunakan RCL kedalam WPF project, ikuti langkah-langkah dibawah ini.

- Tambahkan sebuah Windows Form project lalu buka CS project file dengan cara double click project yang baru saja kita buat. Setelah terbuka ganti nilai dari "Sdk" pada node Project menjadi "Microsoft.NET.Sdk.Razor". Klik kanan pada project lalu Add > Project Reference dan pilih "Latihan.RCL". Lalu copy folder "wwwroot" pada project BlazorLogin kedalam WPF project.
- Untuk menggunakan RCL pada WinForms kita perlu menggunakan BlazorWebView untuk WinForms atau "Microsoft.AspNetCore.Components.WebView.WindowsForms". Oleh karena itu kita perlu menambahkan library tersebut kedalam project dengan menggunakan NuGet seperti pada gambar dibawah ini.



- Buka Form1.cs dan tambahkan component WebView yang baru saja kita tambahkan menggunakan NuGet Package diatas. Lalu drag dan drop WebView dari ToolBox ke dalam Form.



Lalu ganti beberapa properties untuk Component diatas, seperti Dock : Fill dan Name : blazorWebView.

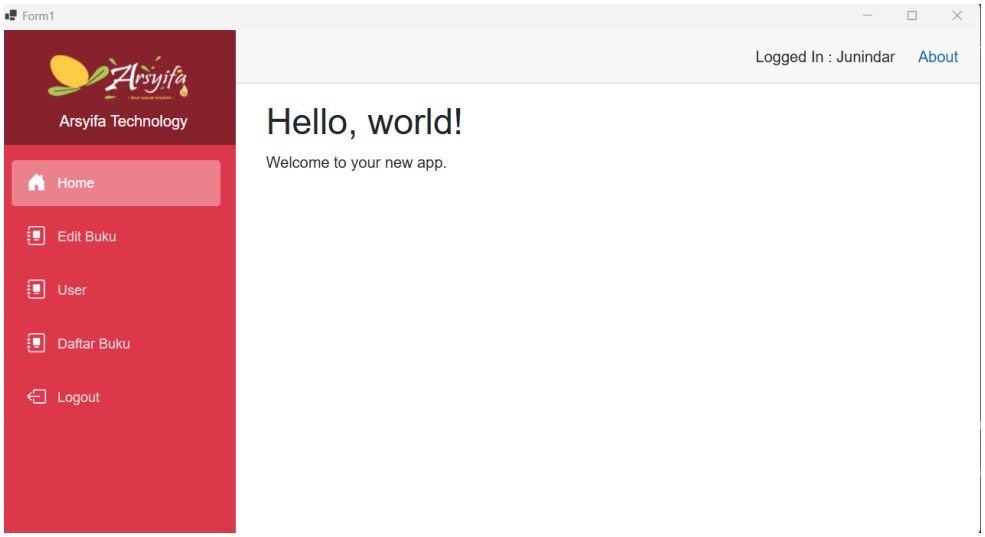
- Klik kanan pada Form1 dan View Code untuk menampilkan jendela Code. Dan ganti sintaks pada file tersebut menjadi seperti dibawah.

```
namespace WindowsFormWithRCL
{
    3 references | 0 changes | 0 authors, 0 changes
    public partial class Form1 : Form
    {
        public Form1(ServiceProvider serviceProvider)
        {
            InitializeComponent();
            blazorWebView.HostPage = "wwwroot/index.html";
            blazorWebView.Services = serviceProvider;
            blazorWebView.RootComponents.Add(
                new RootComponent("#app", typeof(Routes), null));
        }
    }
}
```

- Dan terakhir buka file Program.cs dan ketikkan seperti pada sintaks dibawah.

```
        static void Main()
        {
            // To customize application configuration such as set high DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            var services = new ServiceCollection();
            services.AddWindowsFormsBlazorWebView();
#if DEBUG
            services.AddBlazorWebViewDeveloperTools();
#endif


            services.AddAuthorizationCore();

            services.AddScoped<AuthenticationStateProvider, CustomAuthStateProvider>();
            services.AddTransient<IBookService, BookService>();
            var serviceProvider = services.BuildServiceProvider();
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1(serviceProvider));
        }
    }
```

Jalankan program dan pastikan mendapatkan hasil seperti pada gambar dibawah ini.



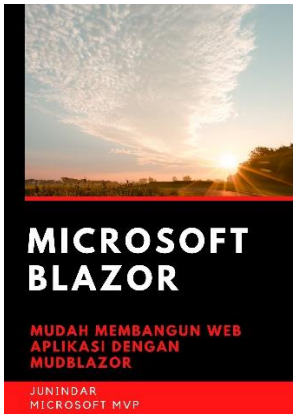
Penutup

Sedangkan untuk memudahkan dalam memahami isi artikel, maka penulis juga menyertakan dengan full source code project latihan ini, dan dapat di download disini https://junindar.blogspot.com/2025/03/menggunakan-blazor-hybrid-pada-wpf-dan.html

Referensi

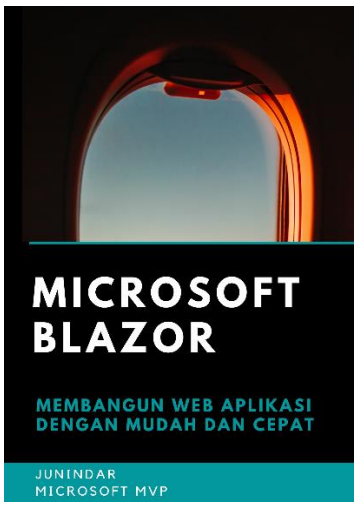
https://play.google.com/store/books/details?id=SH9LEQAAQBAJ&pli=1

https://play.google.com/store/books/details?id=Xc73EAAAQBAJ

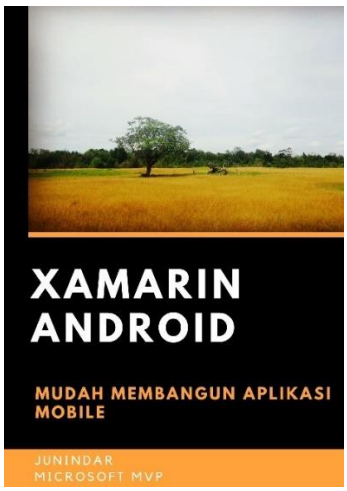


https://play.google.com/store/books/details?id=q9usEAAAQBAJ

Blazor Hybrid pada WPF dan Windows Forms
Junindar, ST, MCPD, MOS, MCT, MVP .NET



https://play.google.com/store/books/details?id=HKZhEAAAQBAJ

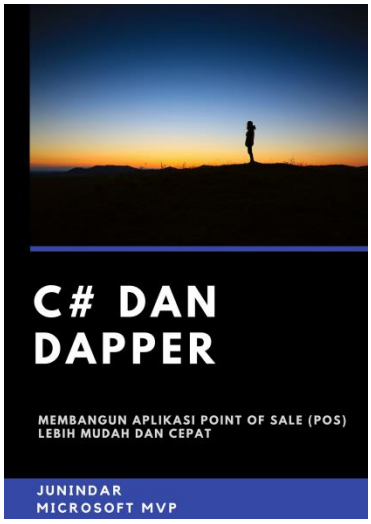


https://play.google.com/store/books/details?id=G4tFDgAAQBAJ

https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

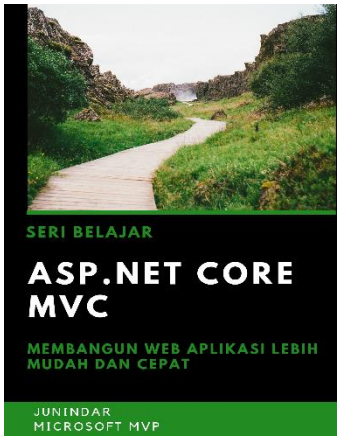
Blazor Hybrid pada WPF dan Windows Forms
Junindar, ST, MCPD, MOS, MCT, MVP .NET

https://play.google.com/store/books/details/Junindar_Xamarin_Forms?id=6Wg-DwAAQBAJ



https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

https://play.google.com/store/books/details/Junindar_ASP_NET_MVC_Membangun_Aplikasi_Web_Lebih?id=XLlyDwAAQBAJ



https://play.google.com/store/books/details/Junindar_ASP_NET_CORE_MVC?id=xEe5DwAAQBAJ

Biografi Penulis.

Junindar Lahir di Tanjung Pinang, 21 Juni 1982. Menyelesaikan Program S1 pada jurusan Teknik Inscreenatika di Sekolah Tinggi Sains dan Teknologi Indonesia (ST-INTEN-Bandung). Junindar mendapatkan Award Microsoft MVP VB pertanggal 1 oktober 2009 hingga saat ini. Senang mengutak-atik computer yang berkaitan dengan bahasa pemrograman. Keahlian, sedikit mengerti beberapa bahasa pemrograman seperti : VB.Net, C#, SharePoint, ASP.NET, VBA. Reporting: Crystal Report dan Report Builder. Database: MS Access, MY SQL dan SQL Server. Simulation / Modeling Packages: Visio Enterprise, Rational Rose dan Power Designer. Dan senang bermain gitar, karena untuk bisa menjadi pemain gitar dan seorang programmer sama-sama membutuhkan seni. Pada saat ini bekerja di salah satu Perusahaan Consulting dan Project Management di Malaysia sebagai Senior Consultant. Memiliki beberapa sertifikasi dari Microsoft yaitu Microsoft Certified Professional Developer (MCPD – SharePoint 2010), MOS (Microsoft Office Specialist) dan MCT (Microsoft Certified Trainer) Mempunyai moto hidup: "**Jauh lebih baik menjadi Orang Bodoh yang giat belajar, dari pada orang Pintar yang tidak pernah mengimplementasikan ilmunya".**