IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Pengenalan Blazor Hybrid

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

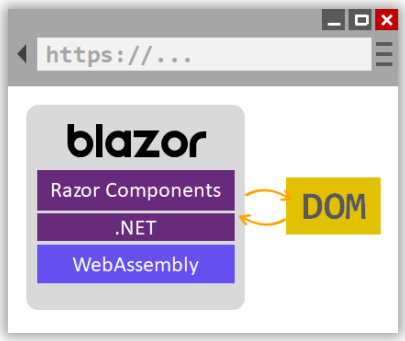
Abstrak

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.

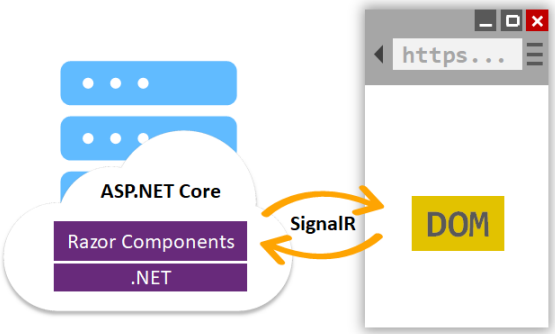
Pendahuluan

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 dirender ke kontrol Web View yang tertanam melalui local interop. Sehingga komponen ini tidak berjalan di browser dan tidak melibatkan WebAssembly.

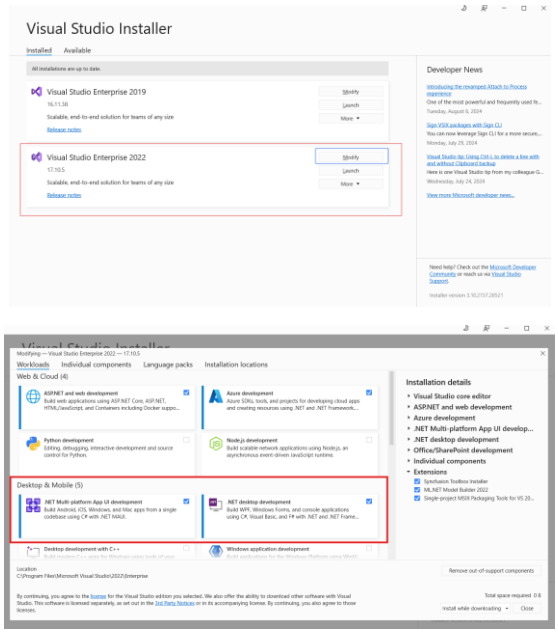
Terdapat beberapa keuntungan bagi pengembang yang ingin membangun aplikasi cross platform dengan menggunakan Blazor Hybrid seperti:

- Kita dapat menggunakan kembali komponen-komponen UI web yang sama di berbagai platform (seluler, desktop, dan web), sehingga mengurangi kebutuhan untuk menulis kode terpisah untuk setiap platform.
- Pengembang dapat menggunakan teknologi web yang sudah dikenal seperti HTML, CSS, dan C# untuk membangun aplikasi, yang dapat mempercepat proses pengembangan.
- Karena komponen Blazor Hybrid berjalan secara native di perangkat, komponen tersebut dapat menawarkan performa yang lebih baik dibandingkan aplikasi web tradisional yang dijalankan di browser.

- Blazor Hybrid memungkinkan kita untuk mengakses fitur dan API perangkat asli melalui .NET MAUI, sehingga dapat membuat aplikasi yang lebih canggih dan kaya akan fitur.

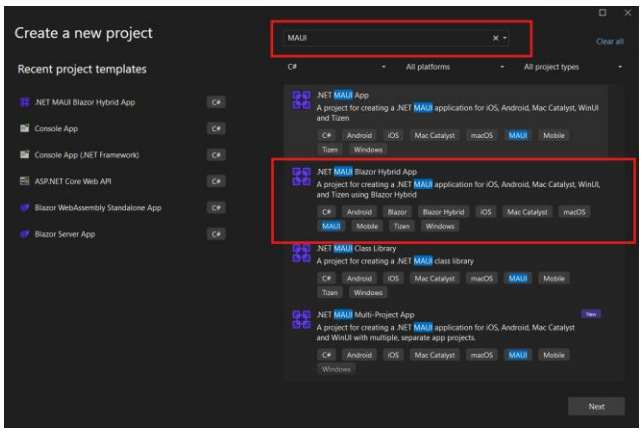
Setup Environment

Untuk membangun aplikasi menggunakan .NET MAUI kita harus pastikan pada Visual Studio sudah terinstall "NET Multi-platform App UI development", sedangkan untuk membangun aplikasi desktop seperti WPF dan Windows Forms yang kita perlukan adalah " .NET desktop development". Untuk menambahkan dua *workload* diatas, gunakan Visual Studio Installer lalu tambahkan workload seperti yang telah disebutkan diatas.

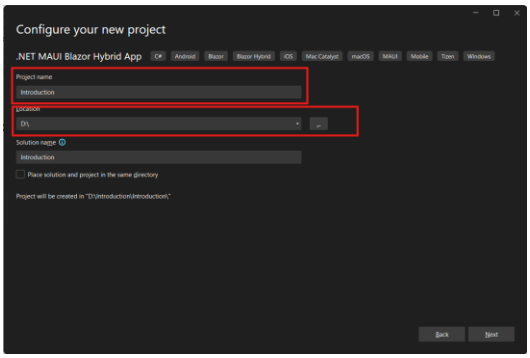


Membuat .NET MAUI Blazor Hybrid App Project

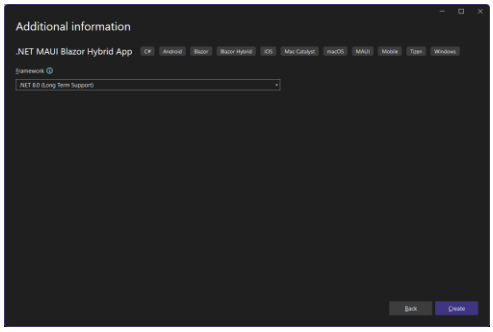
Untuk membuat project Blazor Hybrid, buka Visual Studio lalu klik File > New Project. Selanjutnya pada textbox search, ketikkan "MAUI", sehingga akan menampilkan template-template project yang berkaitan dengan .NET MAUI. Pilih ".NET MAUI Blazor Hybrid App" template dan klik "Next" button.



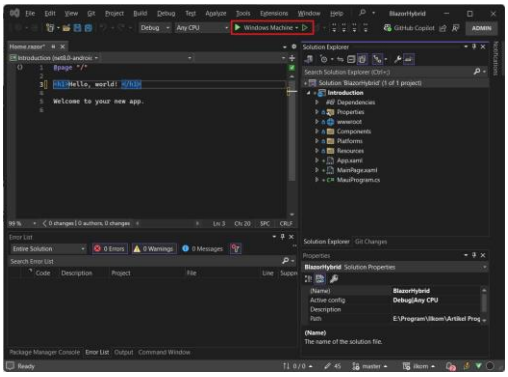
Selanjutnya pada “Configure your new project” dialog, ganti Nama dan Lokasi Project, seperti ada gambar dibawah. Lalu klik “Next” button.



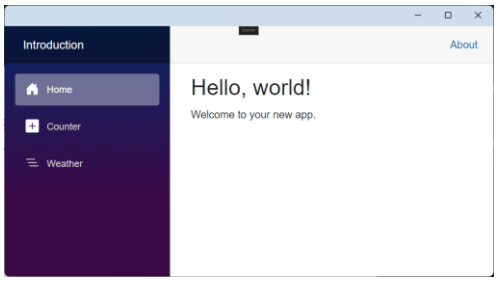
Untuk target framework gunakan .NET 8.0 dan klik “Create” button.



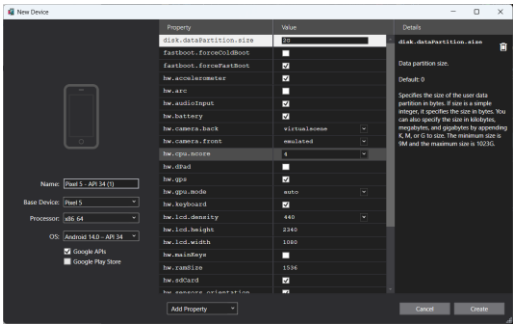
Setelah selesai dengan Langkah-langkah diatas, maka Visual Studio akan membuat sebuah project seperti pada gambar dibawah.



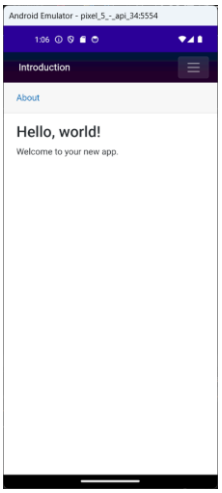
Lalu coba jalankan project dengan menggunakan “Windows Machine” untuk melihat hasilnya. Pada default project terdapat 3 buah menu yaitu Home, Counter dan Weather.



Setelah berhasil menjalankan project pada “Windows Machine”, selanjutnya kita akan mencoba menjalankan project ini dengan menggunakan Android Emulator. Jika belum memiliki Android Emulator di Visual Studio, ikuti Langkah-langkah berikut. Klik Tools > Android > Android Device Manager. Lalu pada Window Android Device Manager klik New button. Lalu muncul window “New Device” seperti dibawah. Pada Latihan ini kita akan membuat emulator menggunakan Base Device-nya adalah Pixel 5 dengan Android 14 dan Size-nya 6 inci.

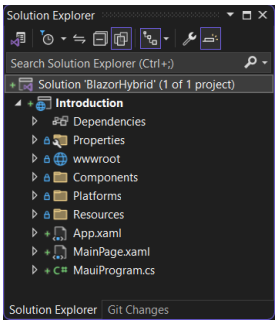


Setelah selesai dengan Langkah-langkah diatas, jalankan emulator yang baru saja dibuat. Lalu kita coba jalankan project dengan menggunakan Android Emulator. Sehingga mendapatkan hasil seperti dibawah.



Explore Project

Pada bagian ini akan dijelaskan bagian-bagian penting pada .NET MAUI Blazor Hybrid App project. Buka solution explorer pada project, disana terdapat beberapa folder dan file yang memiliki fungsi yang berbeda-beda, yang akan dijelaskan seperti dibawah ini.



Wwwroot dan Components Folder : Direktori wwwroot pada aplikasi Blazor Hybrid biasanya digunakan untuk menyimpan file statis seperti CSS, JavaScript, dan gambar. File-file ini dapat diakses secara langsung oleh browser. Namun pada Blazor Hybrid, wwwroot juga berfungsi sebagai titik awal dari aplikasi, di mana file index.html berada. File ini bertindak sebagai host untuk komponen Blazor. Sedangkan Component folder

terdapat file-file seperti pada Blazor project, yaitu MainLayout.razor, NavMenu.razor, _Imports.razor dan lain-lain.

Platforms Folder :

Platforms dalam Blazor Hybrid sangat penting. Pada folder platforms terdapat semua platform yang didukung oleh .NET MAUI seperti Android, iOS, Mac Catalyst, Tyson, and Windows.

Dengan kata lain, folder platforms ini memungkinkan aplikasi Blazor Hybrid dapat berjalan di berbagai platform dengan memanfaatkan kemampuan dari platform tersebut.

Resources Folder :

Resources pada Blazor Hybrid digunakan untuk menyimpan aset statis atau file sumber daya.

Dalam aplikasi Blazor Hybrid, file statis adalah sumber daya aplikasi, diakses oleh komponen Razor menggunakan pendekatan berikut:

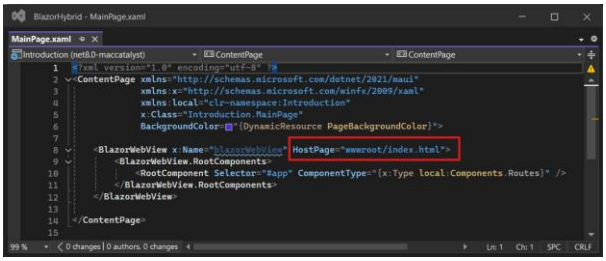
.NET MAUI: menggunakan helper sistem file .NET MAUI.

WPF dan Windows Forms: menggunakan ResourceManager.

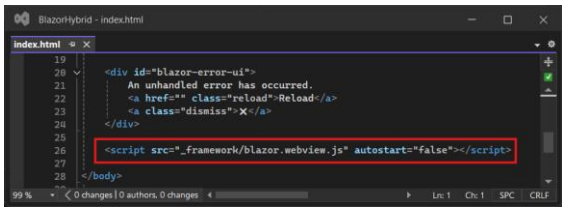
Kita dapat menempatkan aset mentah ke dalam folder Resources/Raw dari aplikasi. Contohnya, kita dapat memiliki file teks statis di Resources/Raw/Data.txt1. Lalu kita dapat membuka file ini dalam komponen Razor dengan memanggil OpenAppPackageFileAsync("Data.txt")1.

Jadi, folder resources ini memungkinkan untuk menyimpan dan mengakses file sumber daya dalam aplikasi Blazor Hybrid.

MainPage.xaml : MainPage.xaml pada Blazor Hybrid berisi BlazorWebView, yang digunakan untuk menampilkan halaman Blazor (.razor) dalam aplikasi. Kita dapat menavigasi ke halaman .razor tertentu dengan mengubah URI yang ditentukan dalam BlazorWebView.



Untuk HostPage telah ditetapkan “index.html” pada wwwroot. Dimana jika kita buka file index.html, akan terdapat javascript “blazor.webview.js” pada script elemen. File ini untuk memastikan komunikasi dengan native webview elemen pada platform yang sesuai.



MauiProgram.cs : File MauiProgram.cs dalam proyek Blazor Hybrid adalah tempat di mana aplikasi dimulai dan konfigurasi awal dilakukan.

Ini adalah tempat di mana kita mendaftarkan layanan yang diperlukan oleh aplikasi dengan Dependency Injection (DI) container. Misalnya, jika menggunakan library seperti Syncfusion Blazor, kita akan mendaftarkan layanan Syncfusion Blazor di file MauiProgram.cs.

Selain itu, file MauiProgram.cs juga digunakan untuk mengatur bagaimana aplikasi berinteraksi dengan sistem operasi. Sebagai contoh, kita dapat menentukan bagaimana aplikasi merespons perubahan orientasi layar, atau bagaimana aplikasi berinteraksi dengan sistem file.

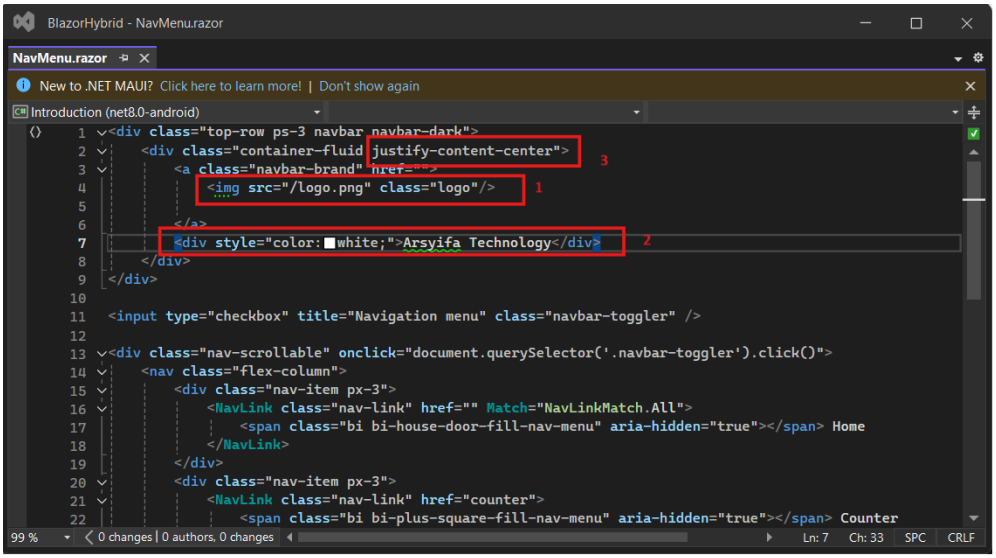
Secara keseluruhan, MauiProgram.cs adalah bagian penting dari proyek Blazor Hybrid kita, dan memahami cara kerjanya dapat membantu kita memanfaatkan sepenuhnya kemampuan Blazor Hybrid.

Mengubah Tampilan pada aplikasi

Pada latihan ini kita akan mencoba untuk mengganti tampilan dari aplikasi. Seperti menambah logo dan mengganti warna background pada aplikasi. Ikuti Langkah-langkah dibawah ini.

- Pertama-tama kita tambahkan sebuah file .png untuk dengan cara klik kanan pada wwwroot lalu pilih Add > Existing Item (Pilih gambar yang akan dijadikan logo pada aplikasi).

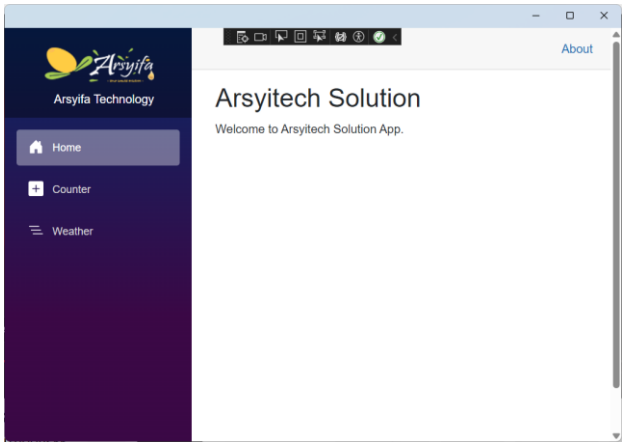
- Lalu buka file “NavMenu.razor” (pada folder Components > Layout) dan kita tambahkan beberapa sintaks seperti berikut. Pertama-tama tambahkan sebuah image dimana menggunakan file yang telah kita tambahkan pada folder wwwroot (1). Kedua tambah sebuah text dengan menggunakan div elemen untuk title dari aplikasi (2). Dan yang terakhir tambahkan css class atribute “justify-content-center” agar posisi image dan text menjadi center.



Lalu buka file” NavMenu.razor.css” dengan cara expand terlebih dahulu file “NavMenu.razor”, dan tambahkan sebuah css class dengan nama “logo”.

```
.logo{
  height:60px;
  display:block;
  margin:auto;
}
```

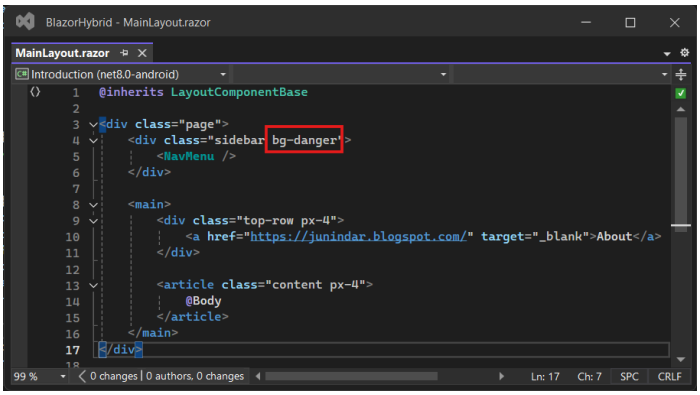
Dilanjutkan dengan mengganti “height” pada “.top-row” menjadi “120px”. Dan jalankan program untuk melihat hasilnya.



- Langkah selanjutnya adalah mengganti warna background dan menghilangkan vertical scrollbar. Bukan file “MainLayout.razor.css”, lalu cari class“.sidebar” dan hapus atau komen sintaks didalamnya.

```
.sidebar {
    /* background-image: linear-gradient(180deg, rgb(5, 39, 103) 0%, #3a0647 70%);*/
}
```

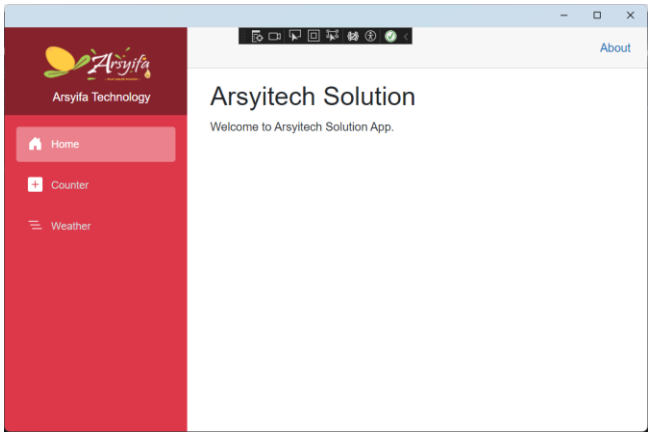
Dan pada “MainLayout.razor” tambahkan css class“bg-danger” seperti pada gambar dibawah.



Dan terakhir untuk menghilangkan vertical scrollbar, pada file “NavMenu.razor.css” >> dan class “.nav-scrollable” ganti nilai height menjadi “120px” (pastikan sama seperti pada class “.top-row” yang telah kita ganti sebelumnya.

```
.nav-scrollable {
    /* Never collapse the sidebar for wide screens */
    display: block;
    /* Allow sidebar to scroll for tall menus */
    height: calc(100vh - 120px);
    overflow-y: auto;
}
```

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



Menambahkan Razor Component pada project

Pada bagian ini kita akan mencoba untuk membuat sebuah Razor component yang berfungsi untuk menampilkan data buku pada project. Sebelum kita memulai hapus terlebih dahulu file Counter.razor dan Weather.razor pada folder Components > Pages. Dan ikuti Langkah-langkah dibawah ini.

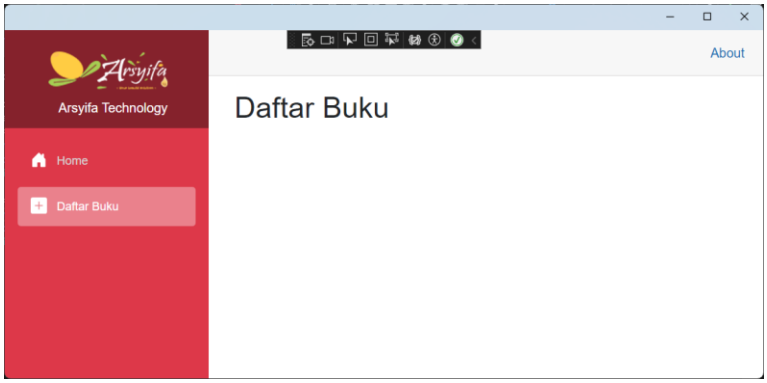
- Tambahkan sebuah Razor Component dengan cara klik kanan pada folder Pages > Add > Razor Component dan ganti Namanya menjadi “BookList.razor”. Dan kita ganti element <h3/> menjadi <h1/> beserta dengan textnya. Selanjutnya kita tambahkan sintaks page directive untuk mengatur route ke komponen yang baru saja kita buat yaitu “/booklist”.

```
@page "/booklist"
<h1>Daftar Buku</h1>

@code {

}
```

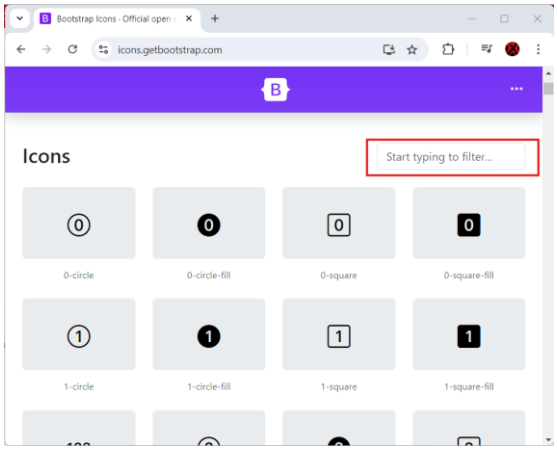
- Selanjutnya buka file “NavMenu.razor”, disana terdapat 3 buah menu. Hapus menu untuk “Counter” , ganti text pada menu “Weather” menjadi “Daftar Buku” dan untuk href (route) ganti menjadi “booklist”.

```
<div class="nav-item px-3">
    <NavLink class="nav-link" href="BookList">
        <span class="bi bi-plus-square-fill-nav-menu" aria-hidden="true"></span> Daftar Buku
    </NavLink>
</div>
```

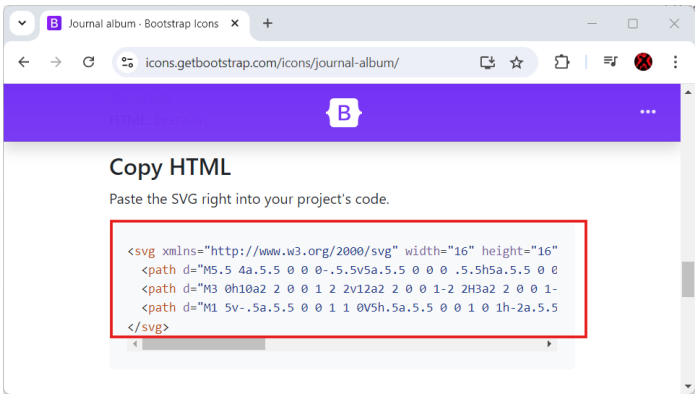
- Jalankan program pastikan menu “Daftar Buku” dapat berjalan dengan baik dan tampilan menjadi seperti dibawah ini.



- Langkah selanjutnya adalah mengganti icon pada menu “Daftar Buku”. Buka file “NavMenu.razor.css” lalu cari css class “.bi-plus-square-fill-nav-menu”. Copy dan ganti namanya menjadi “.bi-list-book-nav-menu”. Dapat kita lihat pada “background-image” property menggunakan svg file. Disini kita terlebih dahulu kita akan membuat svg file untuk mengganti icon pada menu “Daftar Buku”
- Buka https://icons.getbootstrap.com/ pada web browser. Lalu cari icon yang diinginkan, dan klik icon tersebut untuk menampilkan detail informasi.



Selanjutnya copy sintaks HTML seperti pada gambar dibawah untuk dikonversi menjadi css. Untuk mengkonversi svg kita gunakan https://www.svgbackgrounds.com/tools/svg-to-css/ .



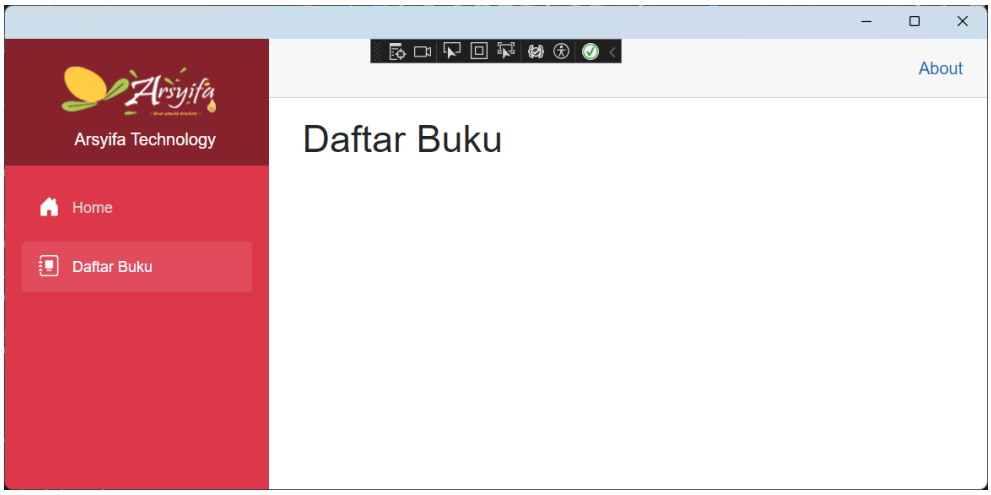
Paste hasil dari konversi kedalam class ".bi-list-book-nav-menu" dan ganti nilai "fill" menjadi white.

```
.bi-list-book-nav-menu {
    background-image: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="white"
}
```

Lalu pada sintaks menu Daftar Buku di NavMenu.razor ubah class pada elemen span menjadi seperti dibawah.

```
<div class="nav-item px-3">
    <NavLink class="nav-link" href="BookList">
        <span class="bi bi-list-book-nav-menu" aria-hidden="true"></span> Daftar Buku
    </NavLink>
</div>
```

Jalankan project dan pastikan mendapatkan hasil seperti dibawah ini.



Membuat Service (Book Service)

Setelah selesai dengan menambahkan razor component pada project, latihan selanjutnya adalah membuat Service (Book Service) untuk memuat data buku. Ikuti Langkah-langkah dibawah ini.

- Buat dua buah folder pada project dengan nama masing-masing “Entity” dan “Service”
- Pada folder “Entity”, tambah sebuah Class dengan nama “Book.cs” dan ketikkan sintaks seperti dibawah.

```
public class Book
{
    public int Id { get; set; }
    public string Judul { get; set; }
    public string Penulis { get; set; }
    public string Deskripsi { get; set; }
    public string Penerbit { get; set; }
    public bool Status { get; set; }
    public string Gambar { get; set; }
}
```

- Tambahkan sebuah Interface dengan nama “IBookService” pada folder “Service” dan ketikkan sintaks seperti dibawah.

```
public interface IBookService
{
  Task<List<Book>?> GetBooksAsync();
}
```

Lalu tambahkan sebuah class “BookService” pada folder Service.

```
public class BookService:IBookService
{
     public async Task<List<Book>?> GetBooksAsync()
     {
         var result = new List<Book>()
         {
             new()
             {
                 Id = 1,
                 Judul = "Visual Studio LightSwitch - Edisi Bundling",
                 Penulis = "Junindar",
                 Penerbit = "EBOOKUID",
                 Deskripsi = "",
                 Status = true,
                 Gambar = "Ls4.jpg"
             },
             //Detail Sintaks ada di project lampiran
         };
        await  Task.Delay(50);
        return result;
     }
 }
```

Method diatas digunakan untuk memuat data Buku, pada latihan ini data buku masih bersifat static (*hardcode*). Pada Class Book terdapat property "Gambar", untuk itu kita perlu menambahkan beberapa image buku kedalam project. Tambahkan folder "images" pada wwwroot, lalu tambahkan beberapa image kedalam folder tersebut.

Inject Service (Book Service) kedalam Component

Setelah selesai membuat BookService seperti yang telah dilakukan diatas, Langkah selanjutnya adalah melakukan Inject Service tersebut kedalam Razor Component. Sebelumnya terlebih dahulu kita perlu register dengan menggunakan Dependency Injection.

Buka file "MauiProgram.cs", sebelum sintaks return statement tambahkan code seperti dibawah, untuk me-register interface IBookService.

```
#if DEBUG
        builder.Services.AddBlazorWebViewDeveloperTools();
        builder.Logging.AddDebug();
#endif
        builder.Services.AddTransient<IBookService, BookService>();

        return builder.Build();
```

Lalu buka file "_Imports.razor" dan tambahkan sintaks dibawah ini. Hal ini kita lakukan agar kita dapat mengakses Service dan Entity pada setiap Razor Component.

```
@using Introduction.Entity
@using Introduction.Service
```

Buka file "BookList.razor", lalu kita inject IBookService seperti gambar dibawah ini.

```
@page "/booklist"
@inject IBookService BookService
<h1>Daftar Buku</h1>
```

Dan pada Code Block tambahkan sintaks berikut.

```
private IEnumerable<Book>? _books = null;
protected override async Task OnInitializedAsync()
{
    _books = await BookService.GetBooksAsync();
}
```

Dari sintaks diatas dapat dilihat, pada method "OnInitializedAsync" kita panggil method "GetBooksAsync" dari BookService. Dimana hasilnya akan disimpan pada "_books" sebuah private field dengan tipenya adalah "IEnumerable<Book>".

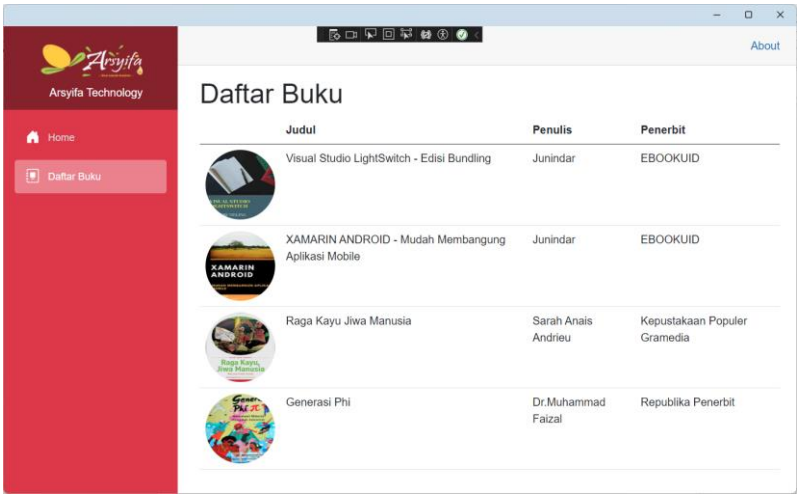
Menampilkan Data pada screen

Pada bagian ini, kita akan menampilkan data buku pada screen yang didapat dari langkah sebelumnya diatas. Disini kita akan mengimplementasikan sintaks untuk UI seperti dibawah.

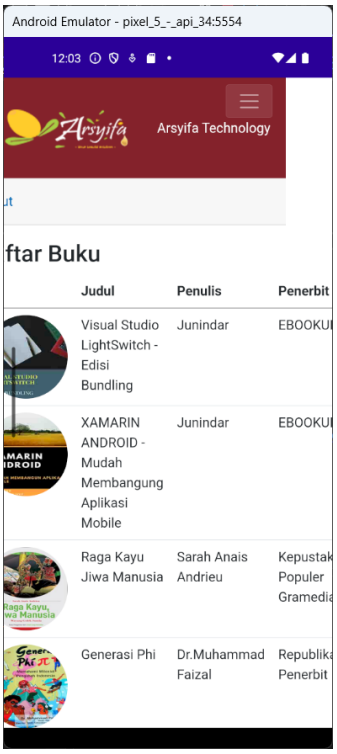
```
@if (_books == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <table class="table">
        <thead>
            <tr>
                <th></th>
                <th>Judul</th>
                <th>Penulis</th>
                <th>Penerbit</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var book in _books)
            {
                <tr>
                    <td>
                    <img src="@($"images/{book.Gambar}")"
                    class="rounded-circle" width="100" height="100" /></td>
                    <td>@book.Judul</td>
                    <td>@book.Penulis</td>
                    <td>@book.Penerbit</td>
                </tr>
            }
        </tbody>
    </table>
}
```

Berikut penjelasan dari sintaks diatas. Pertama-tama jika field “_books” dengan null maka kita akan menampilkan text “Loading”, dan jika memiliki data maka akan kita tampilkan data-data tersebut menggunakan HTML Table. Terdapat 4 buah Header (<th>) yang pertama digunakan untuk gambar buku sehingga kita tidak perlu text pada header tersebut dan dilanjutkan dengan Judul, Penulis dan Penerbit.Sedangkan untuk menampilkan detail data, kita gunakan “foreach” didalam elemen <tbody>.

Jalankan program dengan menggunakan Windows Machine, untuk melihat hasilnya dan pastikan mendapatkan hasil seperti pada gambar dibawah.



Lalu coba jalankan program dengan menggunakan Android Emulator, dan jika sudah terbuka page Daftar Buku coba geser ke kiri atau kekanan. Maka kolom penerbit berada diluar, dan untuk menu akan terdapat “white space” dan screen.



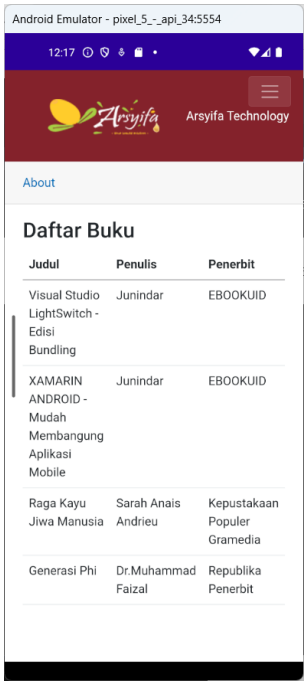
Agar tampilan pada mobile tidak seperti diatas, kita perlu menambahkan sintaks css seperti berikut.

```
<style>
    @@media (max-width:960px){
        .hide-col {
            display: none
        }
    }
</style>
```

Dan pada “<th>” gambar tambahkan class “.hide-col” yang telah kita buat diatas. Lalu lakukan juga pada <tr> untuk menampilkan gambar.

```
<table class="table">
    <thead>
        <tr>
            <th class="hide-col"></th>
            <th>Judul</th>
            <th>Penulis</th>
            <th>Penerbit</th>
        </tr>
    </thead>
```

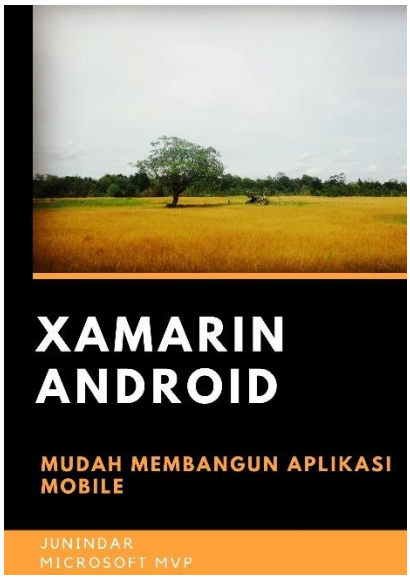
Class css diatas digunakan untuk menyembunyikan kolom gambar jika width dari screen-nya lebih kecil atau sama dengan 960px. Lalu coba jalankan program kembali dengan menggunakan Android Emulator. Pastikan mendapatkan hasil seperti dibawah ini.



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/2024/12/pengenalan-blazor-hybrid.html

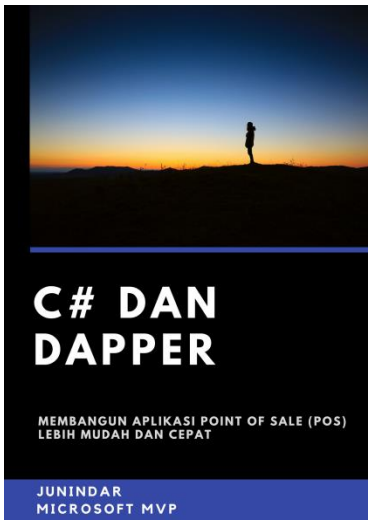
Referensi



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

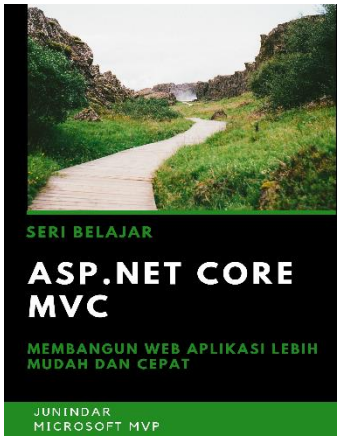
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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".**