IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

MudBlazor Component Pada Blazor – Part 1

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

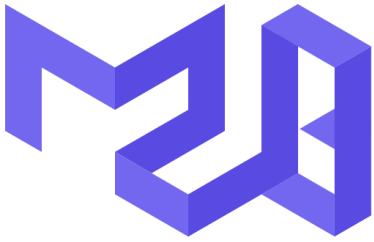
Abstrak

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 WebAssembly, yang artinya kita dapat menjalankan “NET“ didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.

Pendahuluan

MudBlazor adalah sebuah material design component framework yang dibangun khusus untuk Blazor (https://mudblazor.com/). Terdapat banyak komponen pada MudBlazor seperti chart, grid dan lain-lain untuk membantu dalam membangun aplikasi web dengan menggunakan blazor. Seluruh komponen pada MudBlazor dibangun dengan menggunakan C# tanpa javascript kecuali jika sangat diperlukan. Dan dokumentasi untuk penggunaan MudBlazor ini sangat lengkap sehingga membantu para developer untuk menggunakannya.



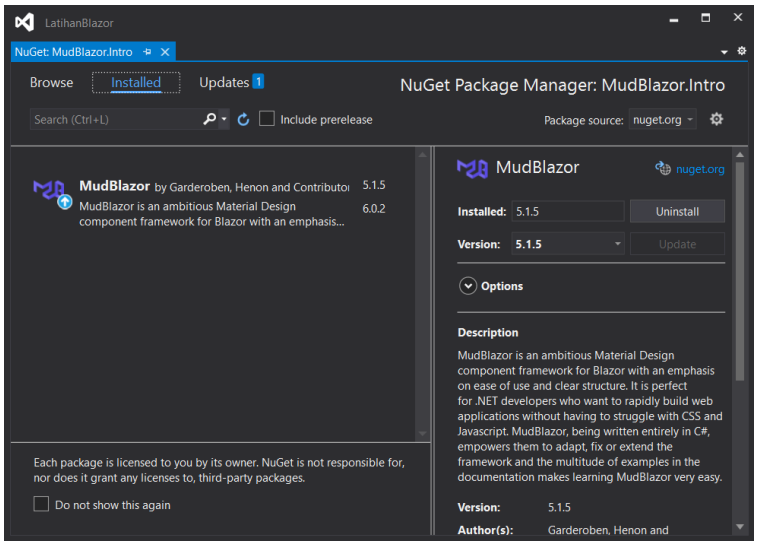
Blazor Components
For faster and easier web development

Pada artikel ini akan dijelaskan bagaimana menggunakan MudBlazor pada sebuah project, dari bagaimana cara install MudBlazor sampai dengan melakukan kustomisasi MudBlazor Theme (MudTheme).

Untuk memudahkan dalam memahami artikel ini, kita buat terlebih dahulu sebuah project Blazor App lalu ikuti langkah-langkah dibawah ini.

- Install MudBlazor pada project.

  Buka Manage Nuget Packages dan ketikkan “MudBlazor”, lalu tekan button Install.



- Konfigurasi MudBlazor

  Setelah package berhasil di install, langkah selanjutnya adalah melakukan konfigurasi. Buka file startup.cs lalu tambahkan MudBlazor service (MudServices) pada method ConfigureServices.

```
services.AddMudServices();
```

  Lalu buka file _Host.cshtml pada file ini akan kita tambahkan reference untuk CSS dan js file dari MudBlazor.

```
<link href="_content/MudBlazor/MudBlazor.min.css" rel="stylesheet" />
<script src="_content/MudBlazor/MudBlazor.min.js"></script>
```

  Dan pada file _Imports.razor kita perlu meng-import package MudBlazor pada project, dengan menambahkan sintaks seperti dibawah ini.

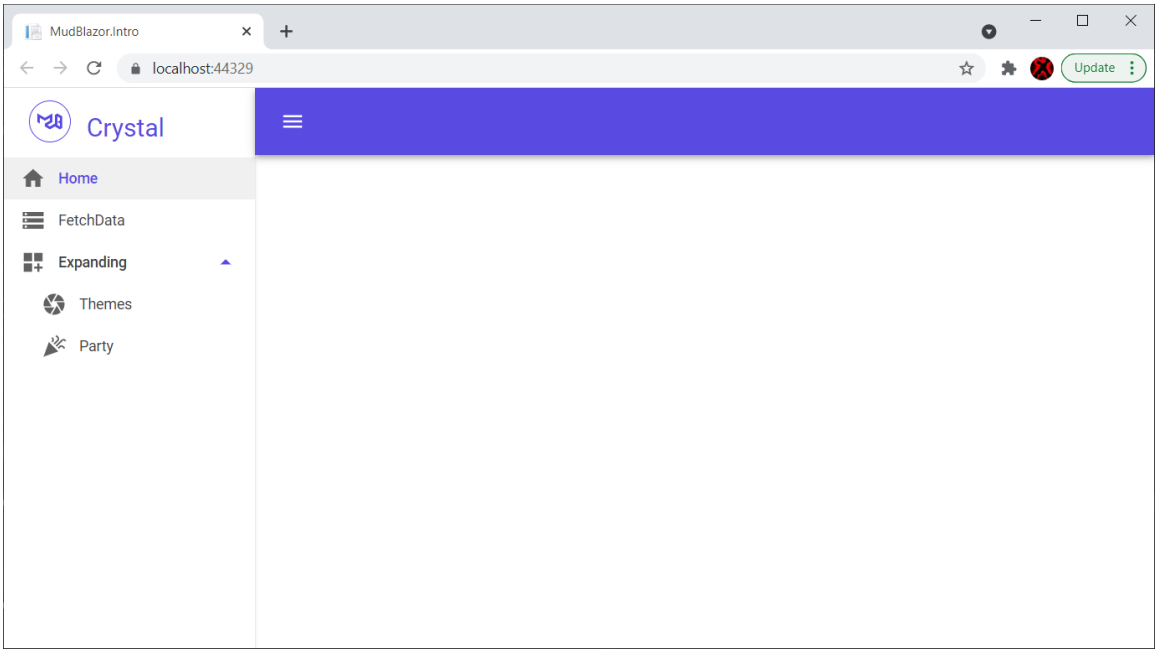
```
@using MudBlazor
```

- Setelah selesai melakukan konfigurasi MudBlazor pada project, kita lanjutkan dengan mengganti sintaks pada MainLayout.Razor dan NavMenu.Razor. Ketikkan sintaks seperti dibawah ini pada MainLayout.Razor. sedangkan untuk sintaks pada NavMenu.Razor dapat dilihat pada project lampiran.

```
<MudThemeProvider />
<MudLayout>
    <MudAppBar>
        <MudIconButton Icon="@Icons.Material.Filled.Menu"
                       Color="Color.Inherit" Edge="Edge.Start"
                       OnClick="@((e) => DrawerToggle())" />

    </MudAppBar>
    <MudDrawer @bind-Open="@_drawerOpen">
        <NavMenu />
    </MudDrawer>
    <MudMainContent>
        <MudContainer MaxWidth="MaxWidth.Medium">
            @Body
        </MudContainer>
    </MudMainContent>
</MudLayout>
@code {

    bool _drawerOpen = true;

    void DrawerToggle()
    {
        _drawerOpen = !_drawerOpen;
    }

}
```

Pada MainLayout terdapat "MudThemeProvider" yang nantinya akan kita kustomisasi sehingga pada aplikasi ini terdapat beberapa theme, yang nantinya pengguna dapat memilih "theme" yang akan digunakan.

Sedangkan pada MudLayout terdapat beberapa section, Seperti MudAppBar, MudDrawer dan MudMainContent. Lalu jalankan program untuk melihat hasilnya seperti pada gambar dibawah ini.



- Langkah selanjutnya adalah melakukan kustomisasi theme.

  Ganti sintaks pada MudThemeProvider seperti dibawah ini.

```
<MudThemeProvider Theme="_currentTheme" />
```

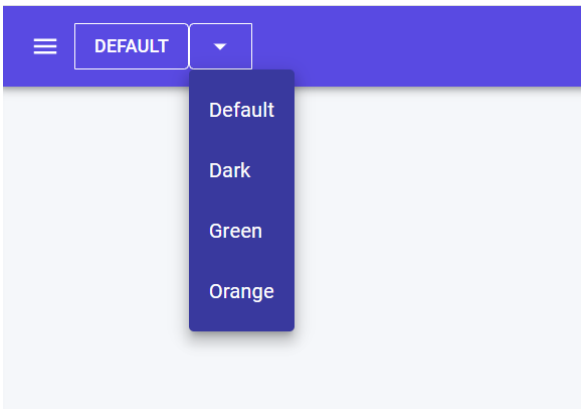
  Dilanjutkan dengan menambahkan sintaks berikut didalam node MudAppBar (dibawah MudIconButton)

```
<MudButtonGroup Variant="Variant.Outlined">
    <MudButton>@_buttonText</MudButton>
            <MudMenu Icon="@Icons.Material.Filled.ArrowDropDown"
             Direction="Direction.Bottom" OffsetY="true">
                <MudMenuItem OnClick="() => SetButtonText(0)">Default</MudMenuItem>
                <MudMenuItem OnClick="() => SetButtonText(1)">Dark</MudMenuItem>
                <MudMenuItem OnClick="() => SetButtonText(2)">Green</MudMenuItem>
                <MudMenuItem OnClick="() => SetButtonText(3)">Orange</MudMenuItem>
            </MudMenu>
</MudButtonGroup>
```

  Dan pada Code Block, kita tambahkan dua buah field dan satu buah method. Field pertama adalah _buttonText (string) yang digunakan untuk Text pada MudButton, sedangkan untuk field kedua adalah _currentTheme dari MudTheme, yang digunakan pada MudThemeProvider. Untuk "SetButtonText" adalah method yang dieksekusi pada saat pengguna memilih theme.

```
private string _buttonText = "Default";
private MudTheme _currentTheme = new MudTheme();

private async Task SetButtonText(int id)
{

}
```

  Lalu jalankan program pastikan mendapatkan hasil seperti pada gambar dibawah ini.



- Tambahkan 4 buah field (MudTheme) yang merupakan implementasi theme dari pilihan diatas (Default, Dark, Grren dan Orange).

```
private MudTheme defaultTheme = new MudTheme()
{
        Palette = new Palette()
        {
            AppbarText = "#ffffff",
            ActionDefault = "#ffffff",
            TextPrimary = "#ffffff",
            Surface = "#39399e",
            Background = "#f5f7fa"
        }
};

Private MudTheme darkTheme = new MudTheme()
{
        Palette = new Palette()
        {
        }
};


Private   MudTheme greenTheme = new MudTheme()
{
        Palette = new Palette()
        {
        }
};

private  MudTheme orangeTheme = new MudTheme()
{
        Palette = new Palette()
        {
        }
};
```

Untuk detail sintaks diatas dapat dilihat pada project lampiran. Palette digunakan untuk mengatur warna yang digunaan pada theme untuk seluruh komponen dan main layout.

```
switch (id)
{
    case 0:
            _currentTheme = defaultTheme;
             _buttonText = "Default";
             break;
    case 1:
            _currentTheme = darkTheme;
            _buttonText = "Dark";
           break;
    case 2:
           _currentTheme = greenTheme;
           _buttonText = "Green";
           break;
    case 3:
          _currentTheme = orangeTheme;
           _buttonText = "Orange";
          break;
}
```

Tambahkan sintaks diatas pada method SetButtonText. Pada MudMenuItem (OnClick) setiap item memiliki parameter yang berbeda-beda, seperti untuk default parameternya adalah 0, Dark 2 sampai pada Orange dengan parameternya adalah 3.

Lalu jalankan program untuk melihat hasilnya, pastikan pada saat theme dipilih maka tampilan pada page berubah juga, seperti pada gambar dibawah ini.

Sampai disini kita telah berhasil melakukan kustomisasi theme pada MudBlazor. Lalu bagaimana jika kita menutup browser dan membukanya kembali? Apakah theme yang keluar adalah yang terakhir kali kita pilih? Jawabannya, theme akan kembali menjadi default. Hal ini dikarenakan kita tidak menyimpan theme yang telah kita pilih.

Untuk latihan ini kita akan menggunakan Web Storage (local storage) untuk media penyimpanannya. Pastikan membaca artikel berikut (http://junindar.blogspot.com/2021/11/javascript-interop-pada-aplikasi-blazor.html) untuk memahami konsep localstorage.

Tambahkan sintaks berikut pada method SetButtonText. Sintaks ini digunakan untuk menyimpan nama theme yang kita pilih.

```
await JsRuntime.InvokeVoidAsync("localStorage.setItem", "theme", _buttonText);
```

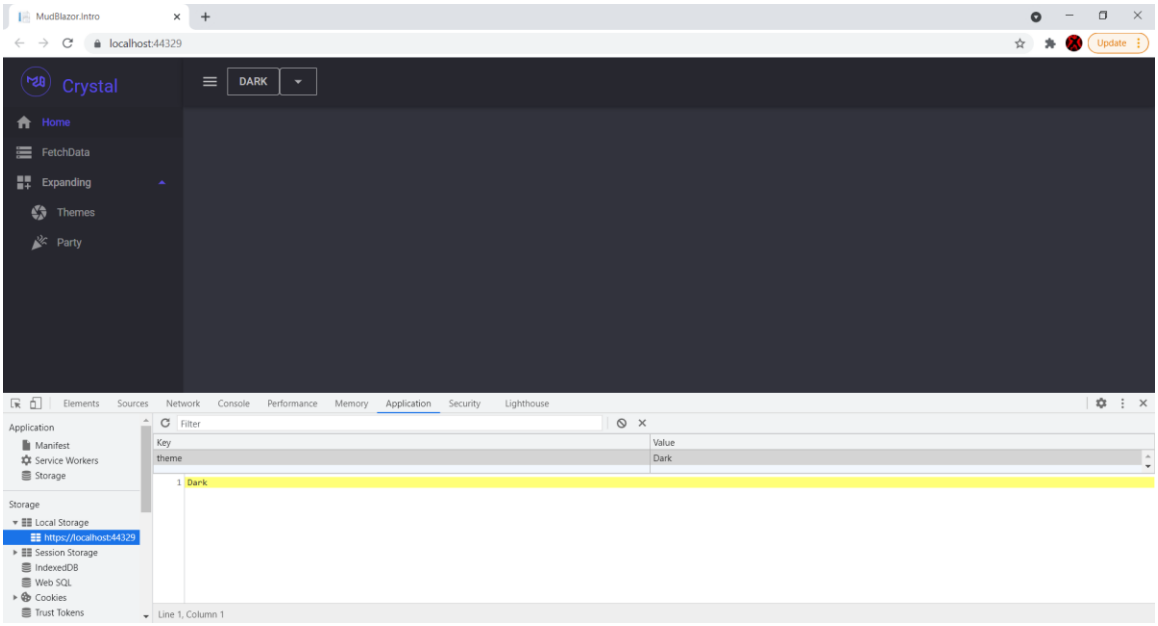
Selanjutnya kita buat sebuah method (InitializeThemeFromLocalStorageAsync) untuk mengambil nama theme pada localstorage dan mengimplementasikannya pada web.

```
private async Task InitializeThemeFromLocalStorageAsync()
{
        var theme = await JsRuntime.InvokeAsync<string>("localStorage.getItem", "theme");
        if (!string.IsNullOrEmpty(theme))
        {
            if (theme == "Default")
            {
                _currentTheme = defaultTheme;
            }
            else if (theme == "Dark")
            {
                _currentTheme = darkTheme;
            }
            else if (theme == "Green")
            {
                _currentTheme = greenTheme;
            }
            else if (theme == "Orange")
            {
                _currentTheme = orangeTheme;
            }

        }
        else
        {
            _currentTheme = defaultTheme;
        }
        _buttonText = theme;
        StateHasChanged();
}
```

Dan yang terakhir kita override method OnAfterRenderAsync dan OnInitialized seperti dibawah. Pada method OnAfterRenderAsync akan kita panggil method diatas (InitializeThemeFromLocalStorageAsync) jika parameter firstRender adalah true.

```
protected override async Task OnAfterRenderAsync(bool firstRender)
{
    if (firstRender)
    {
          await InitializeThemeFromLocalStorageAsync();
    }
}

protected override void OnInitialized()
{
        _currentTheme = defaultTheme;

}
```

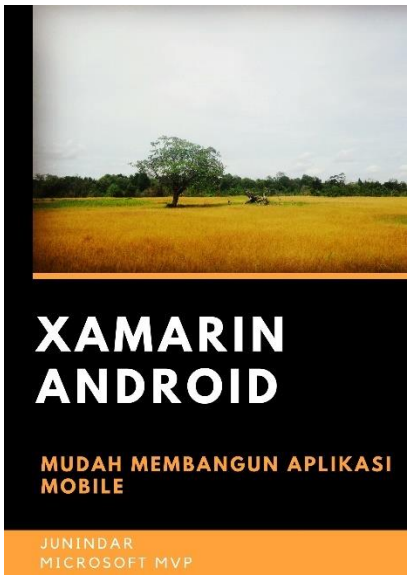
Jalankan program lalu ganti theme dan tutup browser. Selanjutnya jalankan program kembali, pastikan theme yang digunakan adalah theme yang dipilih sebelum browser ditutup.


MudBlazor.Intro
localhost:44329
Update
Crystal
DARK
Home
FetchData
Expanding
Themes
Party
Elements
Sources
Network
Console
Performance
Memory
Application
Security
Lighthouse
Application
Manifest
Service Workers
Storage
Storage
Local Storage
https://localhost:44329
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Filter
Key
Value
theme
Dark
1 Dark
Line 1, Column 1

Penutup

Sedangkan untuk memudahkan dalam memahami isi artikel, maka penulis juga menyertakan dengan full source code project latihan ini, dan dapat di download disini http://junindar.blogspot.com/2021/12/mudblazor-component-pada-blazor-part-1.html

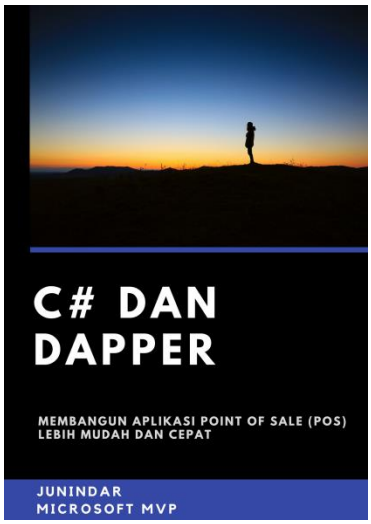
Referensi



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

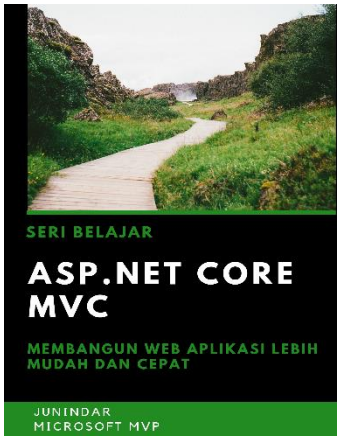
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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

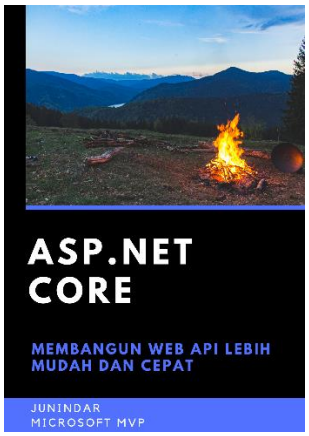


https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

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



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

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".**