IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Pengenalan BLAZOR

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 .

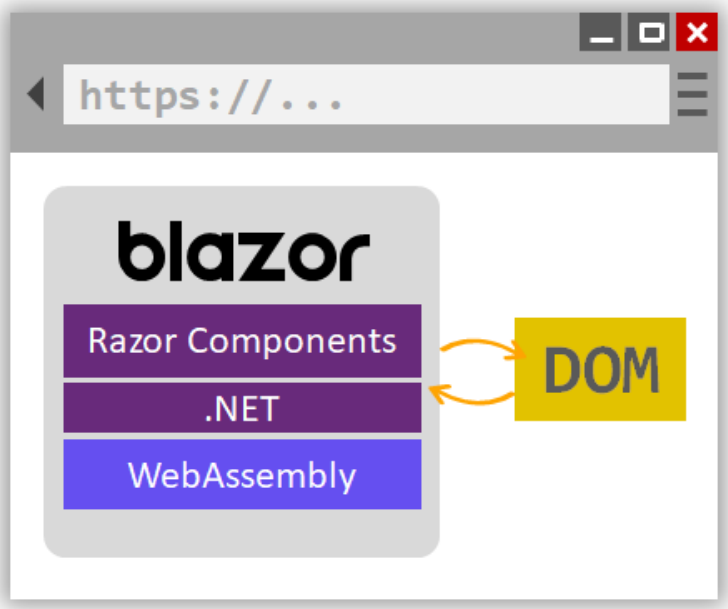
Code .Net berjalan pada WebAssembly, yang artinya kita dapat menjalankan “NET“ didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.

Pendahuluan

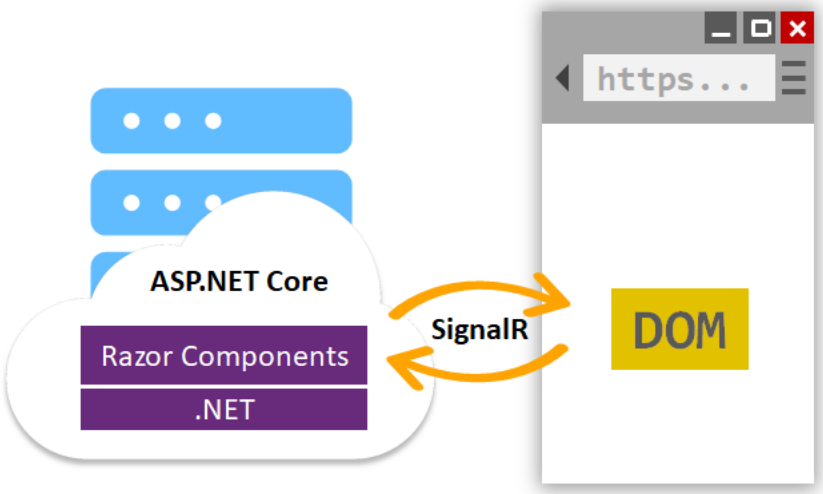
Blazor adalah Web Framework yang bersifat Open Source dimana aplikasi Web yang bersifat client-side interactive dapat dikembangkan dengan menggunakan .Net (C#) dan HTML. Pada saat ini C# biasa digunakan untuk melakukan proses back-end dari aplikasi web. Dengan menggunakan fitur baru dari ASP.NET Core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

Note : Link untuk Project ini ada pada Bab Penutup



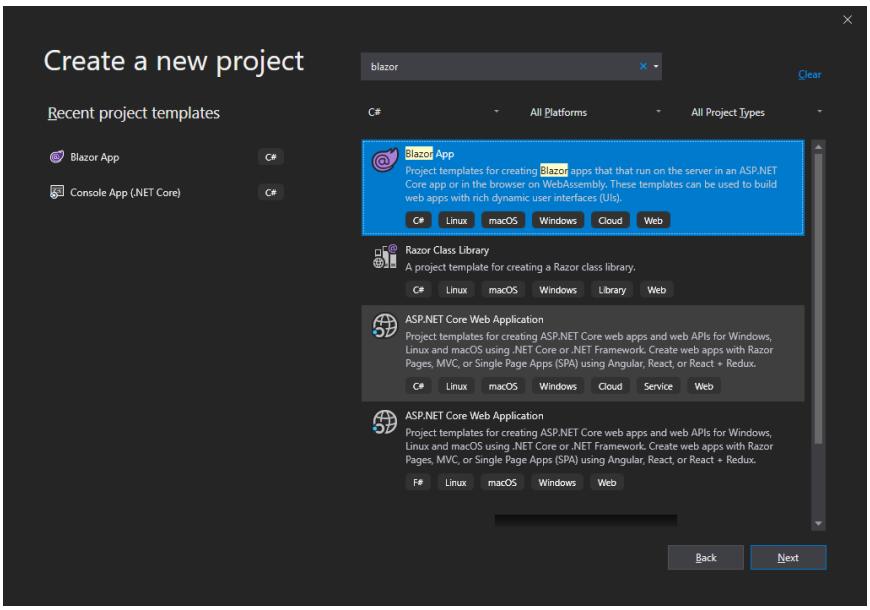
Dengan menggunakan Blazor kita dapat memilih apakah menggunakan WebAssemby (Client Side), seperti yang telah dijelaskan di atas atau Blazor dijalankan diatas server, Pada latihan ini kita akan menggunakan cara kedua yaitu Blazor dijalankan diatas server. Dengan menggunakan cara ini kita memerlukan SignalR 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.



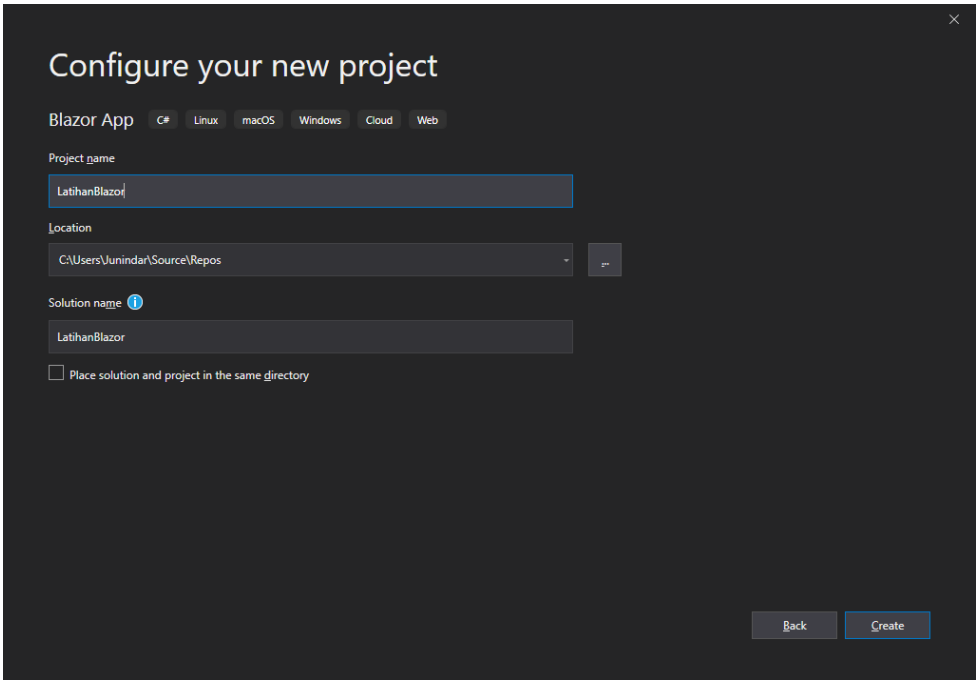
Untuk menggunakan Blazor pastikan pada PC/Laptop sudah terdapat Visual Studio 2019 dan .Net Core 3.0 SDK.

Untuk memudahkan dalam memahami artikel ini, pertama kita buat terlebih dahulu sebuah project Blazor.

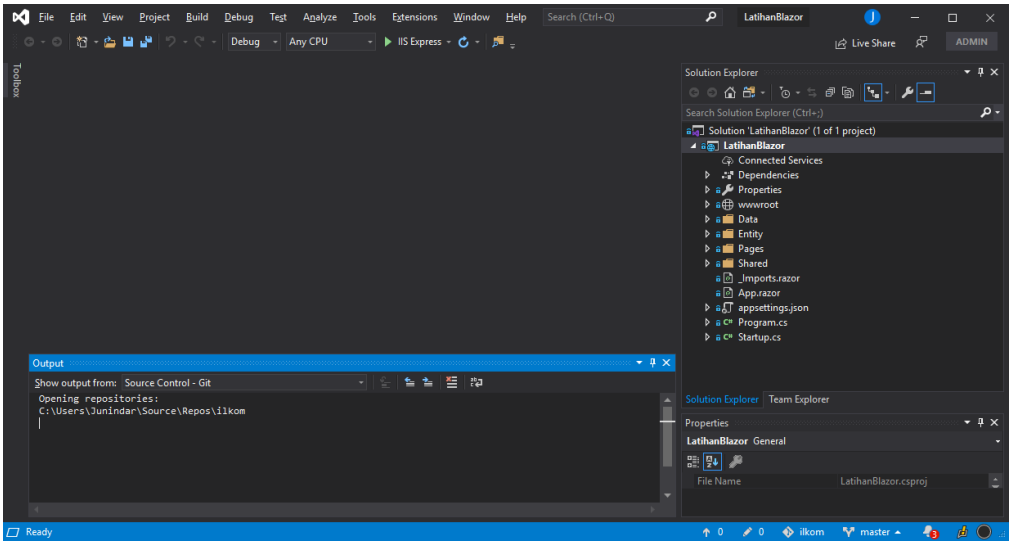
- Buka Visual Studio 2019 dan klik Create New Project
- Pada jendela Create New Project, pilih template Blazor App.



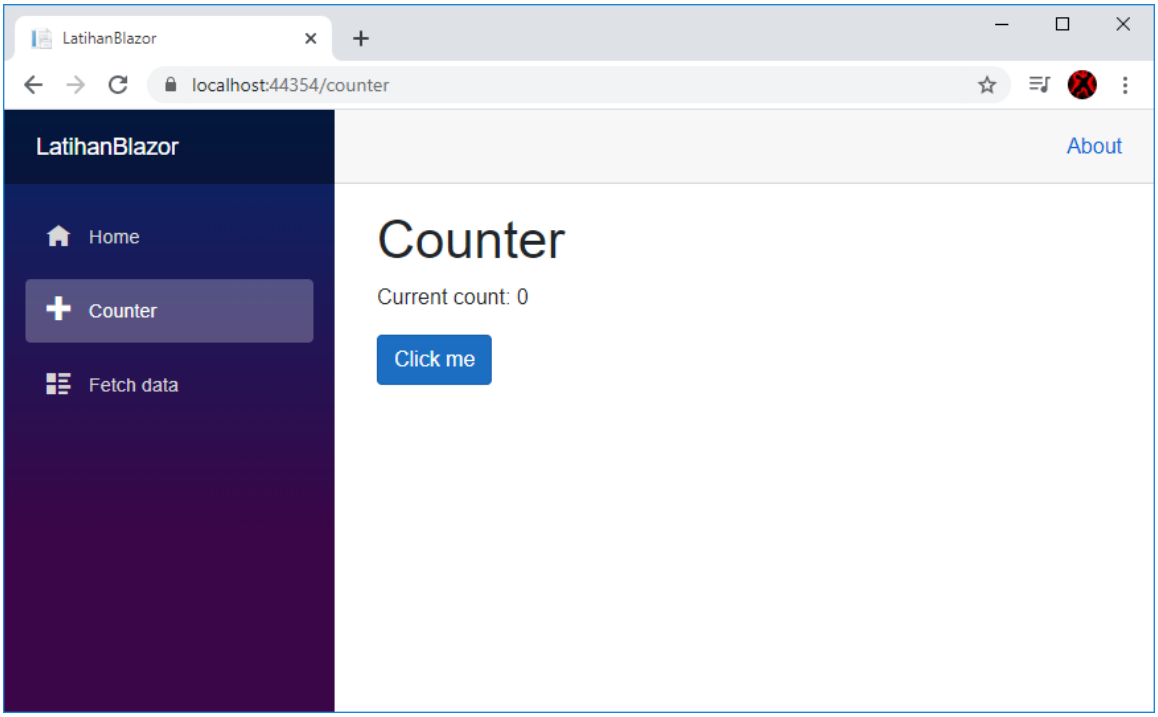
- Klik Next button, selanjutnya ketikkan nama project dan klik Create button.



- Setelah selesai dengan langkah-langkah diatas, Visual Studio akan membuat sebuah project dengan template yang telah kita pilih sebelumnya (Blazor App).



Untuk melihat hasil dari project yang baru saja dibuat jalankan dengan menekan F5. Pada site dibawah, terdapat 3 buah link untuk membuka halaman (page), untuk diketahui halaman-halaman tersebut merupakan Blazor komponen.

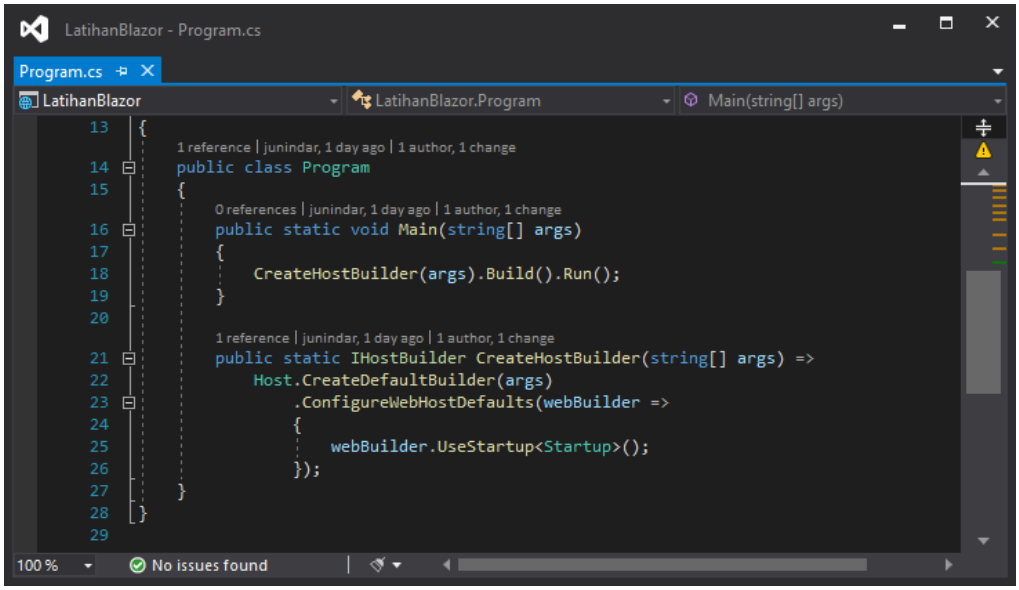


Pada halaman Counter terdapat sebuah button dimana jika kita klik button tersebut maka data pada label Counter count akan berubah, sesuai dengan jumlah klik yang telah kita lakukan. Pada saat kita melakukan klik button “Click me“, maka SignalR akan mengirimkan pesan ke Server dan akan melakukan proses dan setelah selesai SignalR akan mengembalikan hasil dari proses dan mengupdate DOM.

Untuk mengetahui konsep dari Blazor, kita akan melihat struktur project pada Blazor App yang baru saja kita buat. Bagi yang sudah pernah menggunakan ASP.NET Core struktur pada project ini hampir sama dengan ASP.NET Core. Karena Blazor Server Side app, menggunakan ASP.NET Core untuk menjalankannya.

- Program.cs

  Pada saat aplikasi dijalankan, maka pertama kali yang dieksekusi adalah Class Program, dimana pada class tersebut terdapat method “static void Main“ dan dieksekusi diserver side. Didalam Static void Main terdapat “CreateHostBuilder“, yang digunakan untuk melakukan default setup untuk aplikasi. Masih di “CreateWebHostBuilder”, terdapat “UseStartup” dengan mengirimkan class “Startup”. Class Startup sudah ada pada saat kita membuat Project.



- Startup.cs

  Pada Class Startup terdapat dua buah method penting yaitu “ConfigureServices” dan “Configure”. Kedua method ini otomatis dipanggil oleh ASP.NET Core.

  Pada method ConfigureServices kita bisa menambahkan “services” ke dependency injection container melalui “IServiceCollection”. Kita dapat medaftarkan services kedalam aplikasi ASP.NET Core. ASP.NET Core sangat ringan dan juga modular sehingga sangat berbeda dengan ASP.NET versi sebelumnya.

  Pada method Configure terdapat sintaks seperti dibawah.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapBlazorHub();
    endpoints.MapFallbackToPage("/_Host");
});
```

  MapBloazorHub digunakan untuk mengaktifkan koneksi SignalR untuk aplikasi Blazor pada server. Sedangkan pada baris kedua digunakan untuk memastikan semua request menuju ke halaman Host.

- _Host.cshtml

  Jika dilihat halaman ini hanya berisi HTML dan Razor seperti biasa, seperti terdapat beberapa link ke CSS file. Sedangkan pada Body terdapat dua buah bagian penting agar aplikasi Blazor ini dapat berjalan. Yang pertama

Html.RenderComponentAsync, digunakan untuk mengarahkan ke Server Side App. Sedangkan yang kedua adalah blazor.server.js yang merupakan file javascript digunakan untuk memastikan seluruh interaksi antara client dan server berjalan dengan baik.

- App.Razor

  Pada file ini terdapat sebuah tag <Router> yang digunakan untuk mengarahkan kepada halaman yang benar. Didalam tag router terdapat tag Found dan NotFound. Kedua tag tersebut menggunakan MainLayout sebagai DefaultLayout atau Layout.

- MainLayout.Razor

  File ini terletak pada Shared folder, terdapat 2 tag div, yang pertama menggunakan class sidebar sedangkan yang kedua main. Untuk sidebar terdapat NavMenu yang digunakan untuk menampilkan menu, yang detail sintaksnya terdapat pada file NavMenu.Razor. untuk class main terdapat "@Body", yang digunakan untuk razor component.

File-file yang telah dijelaskan diatas merupakan file-file penting yang harus diketahui fungsinya dalam membangun aplikasi dengan menggunakan Blazor Server Side.

Pada default project terdapat sebuah file pada Pages folder yaitu Counter.razor. file ini merupakan razor component. Dimana didalam file ini terdapat sintaks C# dan HTML. Terdapat sebuah code block seperti dibawah yang memiliki sebuah method dengan nama IncreamentCount.

```
@code {
    int currentCount = 0;

    void IncrementCount()
    {
        currentCount++;
    }
}
```

Method ini akan dipanggil pada saat button "Click me" diklik. Pada button terdapat @onclick, yang digunakan untuk memanggil method IncreamentCount.

Pada file ini sintaks HTML dan C# (Code Block) digabung menjadi satu file, tentu jika kita membuat simple komponen menggabungkan HTML dan C# dalam satu file tidak akan menjadi masalah. Tetapi jika komponen yang memiliki fungsi yang bnyak tentu

akan membuat sintaks baik HTML dan C# juga semakin banyak dan tidak disarankan untuk menggabungkan menjadi satu file. Oleh karena itu kita dapat memisahkan sintaks HTML dan C# menjadi file yang berbeda. Untuk sintaks C# kita akan menggunakan class tersendiri dengan menggunakan ComponenBase. Untuk memudahkan ikuti langkah-langkah dibawah ini.

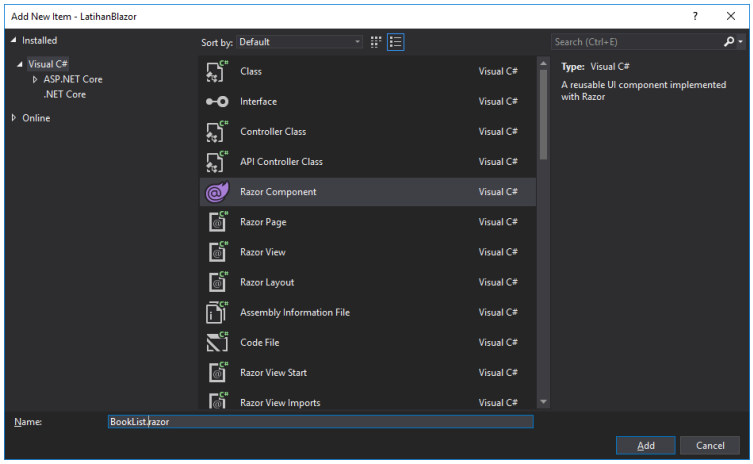
- Buat sebuah folder dengan nama Entity dan tambahkan dua buah class Category dan Book. Ketikkan sintaks dibawah untuk masing-masing class.

```
public class Category
{
        public int CategoryID { get; set; }

        public string NamaCategory { get; set; }

        public List<Book> Books { get; set; }
}


public class Book
{
        public int BookID { get; set; }
        public string Judul { get; set; }
        public string Penulis { get; set; }
        public string Penerbit { get; set; }
        public string Deskripsi { get; set; }
        public bool Status { get; set; }
        public string Gambar { get; set; }
        public int CategoryID { get; set; }
        public Category Category { get; set; }

}
```

- Selanjutnya pada folder Pages tambahkan sebuah class dengan nama "BookListBase". Seperti yang telah dijelaskan diatas, code behind dari razor komponen ini menggunakan (inherits) dari ComponentBase.

  ```
  public class BookListBase : ComponentBase
  ```

  dimana sebelumnya kita harus import pada class ini sebuah namespace "`using Microsoft.AspNetCore.Components;`"
- Lalu tambahkan sebuah razor component dengan nama BookList.razor.




```
@page "/BookList"
@inherits BookListBase
```

Tambahkan sintaks diatas pada file yang baru saja kita buat pada baris paling atas sintaks. Baris pertama digunakan untuk menavigasi ke BookList. Sedangkan pada baris kedua digunakan untuk menghubungkan dengan code behind (BookListBase). Dan hapus code block pada file ini.

- Pada latihan ini, kita akan menggunakan static data dimana datanya kita create langsung pada class (hard code).

```
protected override Task OnInitializedAsync()
{

    InitializeCategories();
    InitializeBooks();

    return base.OnInitializedAsync();
}
```

Disini kita melakukan overide pada methos OnInitializeAsync, terdapat dua buah method yaitu InitializeCategories dan InitializeBooks. Dua buah method diatas digunakan untuk inisialisasi data Category dan Book (Detail sintaks dapat dilihat pada project lampiran). Pada file ini juga terdapat properti `public IEnumerable<Book> Books { get; set; }` , dimana data-data Book akan diisi didalamnya dan nantinya digunakan pada fie BookList.Razor.

- Buka kembali file BookList.razor dan ketikkan sintaks dibawah.

```
@if (Books == null)
{
    <p><em>Data tidak tersedia</em></p>
}
else
{
 //Detail Sintaks pada lampiran
}
```

terdapat dua buah kondisi disini, jika Books sama dengan Null maka page akan terdapat sebuah text "Data tidak tersedia", dan jika ada datanya maka data-data tersebut akan ditampilkan pada HTML table.

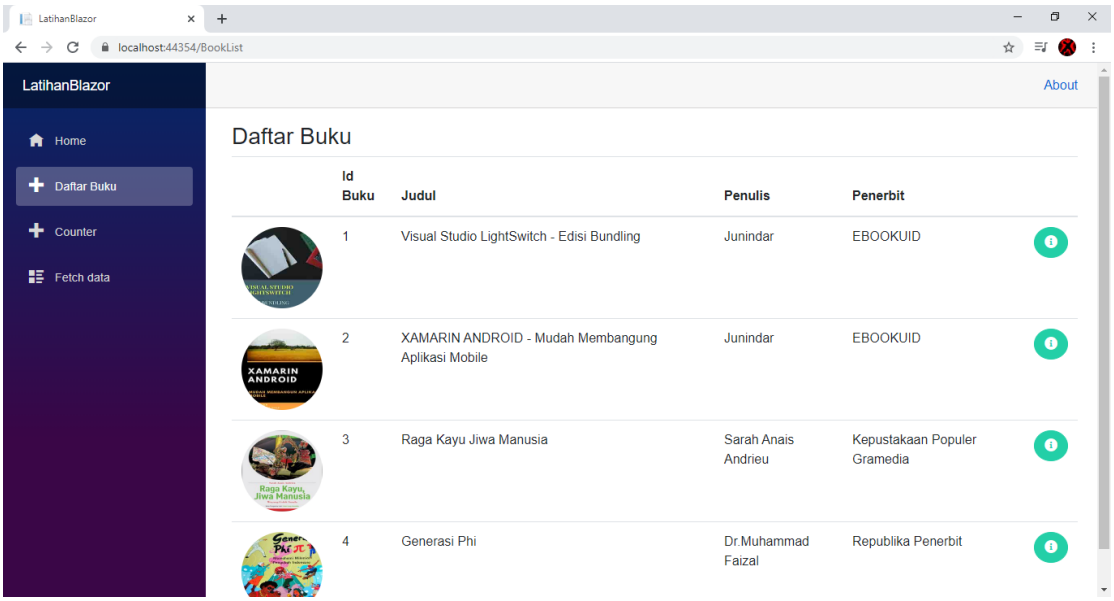
```
<table class="table">
        <thead>
        <tr>
            <th></th>
            <th>Id Buku</th>
            <th>Judul</th>
            <th>Penulis</th>
            <th>Penerbit</th>
            <th></th>
        </tr>
        </thead>
        <tbody>
        @foreach (var book in Books)
        {
            <tr>

                <td><img src="@($"images/{book.Gambar}")" class="rounded-circle"
width="100" height="100" /></td>
                <td>@book.BookID</td>
                <td>@book.Judul</td>
                <td>@book.Penulis</td>
                <td>@book.Penerbit</td>
                <td>
                    <a href="@($"bookdetail/{book.BookID}")" class="btn btn-
primary table-btn">
                        <i class="fas fa-info-circle"></i>
                    </a>
                </td>
            </tr>
        }
        </tbody>
    </table>
```

Sebelumnya pada folder wwwroot, tambahkan sebuah folder images dan copy file gambar dari project lampiran. Pada kolom terakhir kita menggunakan html link dimana fungsi akan digunakan untuk menampilkan detail buku pada halaman yang berbeda. Dimana BookId yang digunakan sebagai paramternya.

Sebelum kita membuat halaman tersebut pastikan halaman ini dapat berjalan dengan baik terlebih dahulu. Lalu tambahkan link menu pada file NavMenu.razor seperti pada sintaks dibawah.

```
<li class="nav-item px-3">
    <NavLink class="nav-link" href="BookList">
          <span class="oi oi-plus" aria-hidden="true"></span> Daftar Buku
    </NavLink>
</li>
```

Dan jalankan project ini pastikan mendapatkan hasil seperti gambar dibawah ini.



- Setelah selesai dengan halaman diatas, selanjutnya kita akan membuat untuk halaman BookDetail. Buat sebuah class dengan nama BookDetailBase pada folder Pages. Untuk detail sintaks dapat dilihat di project lampiran.

```
public class BookDetailBase : ComponentBase
{
        [Parameter]
        public string BookId { get; set; }
        public Book Book { get; set; } = new Book();

        protected override Task OnInitializedAsync()
        {
            InitializeCategories();
            InitializeBooks();

            Book = Books.FirstOrDefault(e => e.BookID == int.Parse(BookId));
            var cat = Categories.First(c => c.CategoryID == Book.CategoryID);
            Book.Category = cat;
            return base.OnInitializedAsync();
        }
}
```

Pada class ini terdapat sebuah variable string BookId, dimana variable ini adalah sebuah parameter, yang nantinya akan dikirimkan melalui razor komponen. Sedangkan property Book digunakan untuk menampung hasil pencarian berdasarkan parameter tersebut.

- Tambahkan sebuah Razor komponen dengan nama BookDetail.

```
@page "/bookdetail/{BookId}"
@inherits BookDetailBase
```

Pada baris pertama dapat kita lihat merupakan navigasi ke bookdetail dengan mengirimkan parameter BookId. Untuk menampilkan hasilnya pada layar dapat dilihat pada sintaks dibawah.

```
<h1 class="page-title">Detail Buku : @Book.Judul</h1>

<div class="col-12 row">
    <div class="col-4">
        <img src="@($"images/{Book.Gambar}")"   width="200" height="300" />
    </div>
    <div class="col-8 row">
        <div class="col-xs-12 col-sm-8">
            <div class="form-group row">
                <label class="col-sm-4 col-form-label">Book ID</label>
                <div class="col-sm-8">
                    <label type="text" class="form-control-
                     plaintext">@Book.BookID</label>
                </div>
            </div>
             @*Detail sintaks ada lampiran project*@
        </div>
    </div>
</div>
```

Pada halaman ini akan dibagi dua yang pertama untuk gambar buku (`<div class="col-4">`), sedangkan yang kedua (`<div class="col-8 row">`) untuk menampilkan data-data buku.

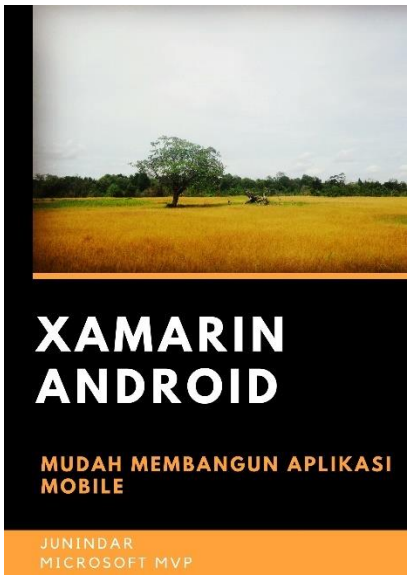
Sekali lagi jalankan program pada halaman daftar buku, klik button detail. Dan pastikan mendapatkan hasil seperti dibawah ini.

Artikel ini akan dilanjuti dengan materi selanjutnya yaitu membuat proses Insert, Delete dan Update menggunakan EF Core dan Blazor. Artikel ini hanya membahas konsep dasar dari Blazor yang harus diketahui sebelum kita membuat aplikasi menggunakan Blazor.

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/2020/02/pengenalan-blazor.html

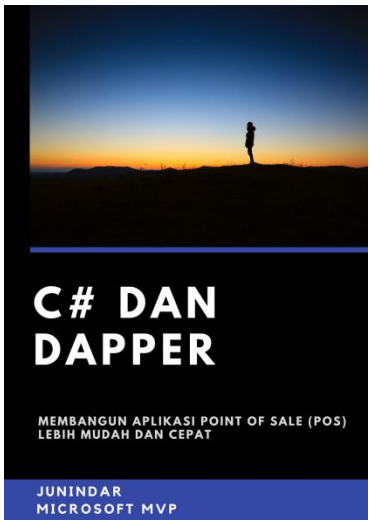
Referensi



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

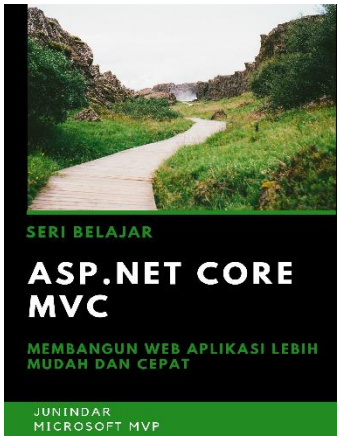
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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".**