IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Dialog Component pada Blazor Hybrid

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com
http://junindar.blogspot.com

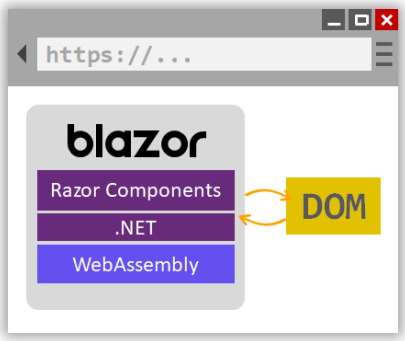
Abstrak

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.

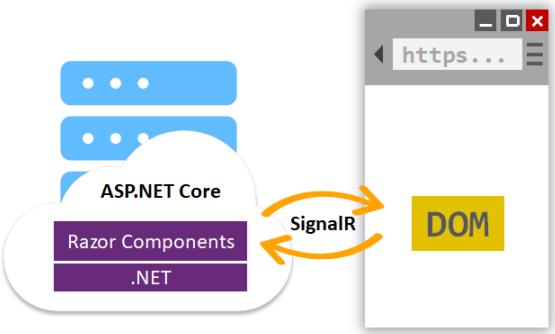
Pendahuluan

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

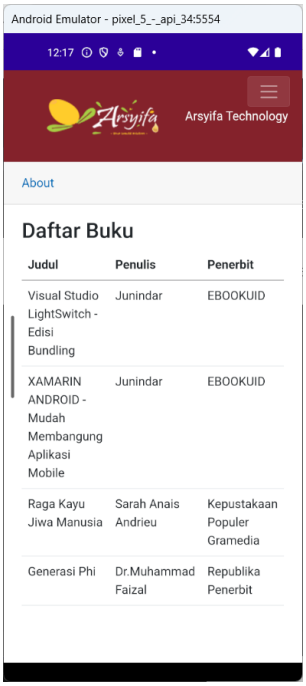
Disarankan untuk membaca dan menyeselesaikan latihan pada artikel sebelumnya pada tautan berikut : https://junindar.blogspot.com/2024/12/pengenalan-blazor-hybrid.html

Pada artikel ini tidak menjelaskan apa itu blazor hybrid, bagaimana bekerja dengan project maupun cara-cara untuk menambahkan item pada project, karena semuanya telah dijelaskan pada dua artikel sebelumnya. Pastikan anda telah menyelesaikan latihan-latihan pada artikel sebelumnya. Artikel ini akan focus bagaimana membuat Dialog Component pada blazor hybrid.

Untuk memulai latihan pada artikel ini buat terlebih dahulu Blazor Hybrid App Project pada Visual Studio 2022.

Note : Link untuk Project Lampiran ada pada Penutup.

Sebelum kita masuk bagaimana membuat Dialog pada blazor hybrid, terlebih dahulu kita akan kembali pada artikel sebelum ini. Pada artikel sebelumnya, kita menggunakan format table untuk menampilkan data pada halaman daftar buku. Seperti pada gambar dibawah ini.



Disini kita akan mengubah tampilan Daftar Buku dari yang sebelumnya berbentuk table menjadi vertikal sehingga lebih jelas pada saat aplikasi dibuka pada *handphone*. Ikuti langkah-langkah dibawah ini.

- Buka file “BookList.razor” lalu tambahkan sintaks css seperti dibawah.

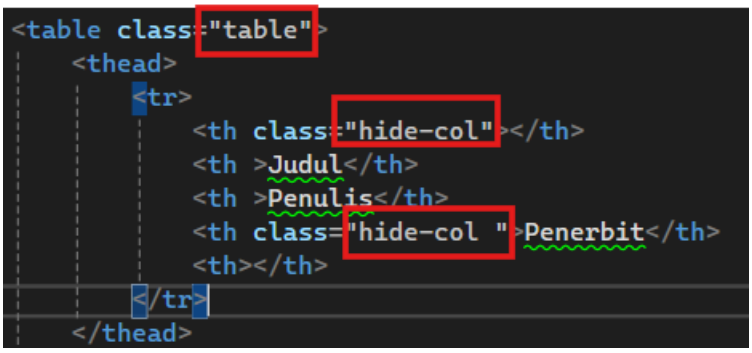
```
table {
     width: 100%;
     border-collapse: collapse;
 }

 th, td {
     padding: 10px;
     text-align: left;
     border: 1px solid #ddd;
 }

 img {
     max-width: 100%;
     height: auto;
     border-radius: 50%;
 }
```

```
@@media (max-width: 600px) {
    table, thead, tbody, th, td, tr {
        display: block;
    }
    th {
        display: none;
    }
    td {
        display: flex;
        flex-direction: column;
        align-items: center;
        padding: 10px;
        border: none;
        border-bottom: 1px solid #ddd;
    }
    td:before {
        content: attr(data-label);
        flex-basis: 100%;
        text-align: center;
        font-weight: bold;
        margin-bottom: 5px;
    }
```

Selanjutnya hapus attribute class pada <table> dan <th> seperti pada gambar dibawah ini.

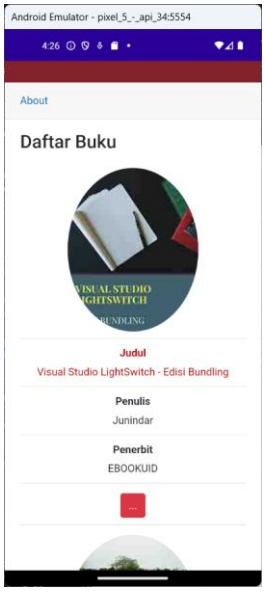


Sedangkan untuk menampilkan data buku detail kita ganti sintaksnya seperti berikut.

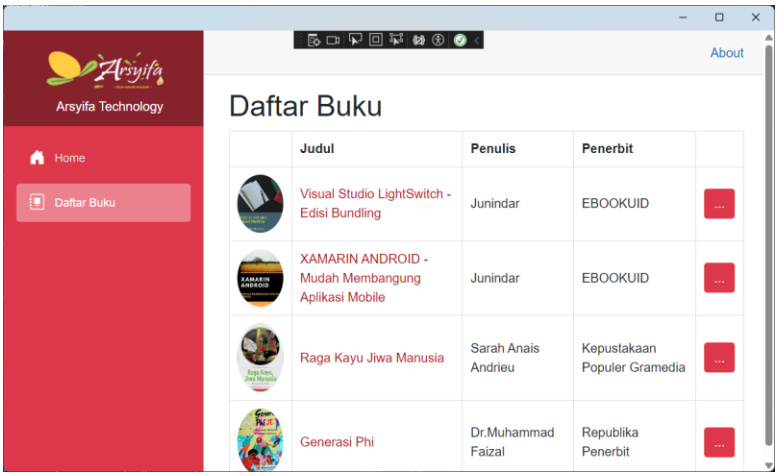
```
@foreach (var book in _books)
{
    <tr>
            <td><img src="@($"images/{book.Gambar}")" /></td>
            <td data-label="Judul" style="color:#B10505;">@book.Judul</td>
            <td data-label="Penulis">@book.Penulis</td>
            <td data-label="Penerbit">@book.Penerbit</td>
        <td>
            <a href="@($"bookdetail/{book.Id}")" class="btn btn-danger table-btn">
                ...
            </a>

        </td>
    </tr>
}
```

Lalu jalankan hasilnya dan pastikan mendapatkan hasil seperti pada gambar dibawah ini



Tampilan pada Mobile



Tampilan pada Windows machine

Dialog

Pembahasan berikutnya adalah membuat halaman dialog pada Blazor Hybrid. Blazor Dialog Component adalah komponen modal popup yang digunakan dalam aplikasi Blazor untuk menampilkan informasi kepada pengguna. Komponen ini dapat juga digunakan untuk meminta input dari pengguna. Sebagai contoh, kita dapat membuat dialog untuk konfirmasi, form input, atau pesan peringatan.

Untuk latihan kali ini kita akan menampilkan data buku dengan menggunakan dialog. Ikuti Langkah-langkah dibawah ini.

- Tambahkan sebuah razor component dengan nama “BookDetailDialog.razor”
- Lalu ketikkan sintaks html seperti dibawah ini

```
@inject IBookService BookService
@if (ShowDialog)
{
    if (Book == null)
    {
        <p><em>Data Tidak Tersedia</em></p>
    }
    else
    {
        <div class="modal fade show d-block" id="exampleModal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content" >
                    <div class="modal-header">
                        <h5 class="modal-title" id="titleLabel">Detail Boku</h5>
                        <button type="button" class="close" data-dismiss="modal"
                         aria-label="Close" @onclick="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">

                        <div class="col-12 row">
                            <div class="col-4 col-12">
                                <img src="@($"images/{Book.Gambar}")" width="200" height="300" />
                            </div>
                            <div class="col-8 col-12 row">
                                <div class="col-xs-12 col-sm-8">
                                    @*Detail sintaks ada lampiran project*@
                                </div>
                            </div>
                        </div>
                        <a class="btn btn-outline-primary edit-btn" @onclick="@Close">Close</a>
                    </div>
                </div>
            </div>
        </div>
    }

}
```

- Selanjutnya tambahkan sintaks C# untuk *back-end* seperti dibawah, untuk penjelasan detail dari sintaks html dan C# akan dijelaskan setelah ini.

```
@code {
    public bool ShowDialog { get; set; }
    public Book? Book { get; set; }
    [Parameter]
    public EventCallback<bool> CloseEventCallback { get; set; }

    public async void Show(int bookid)
    {
        if (bookid >0)
        {
            Book = await BookService.GetBookByIdAsync(bookid);
        }
        ShowDialog = true;
        StateHasChanged();
    }
    public void Close()
    {
        ShowDialog = false;
    }
}
```

Disini akan dijelaskan untuk sintaks C# terlebih dahulu. Terdapat 2 buah property yaitu "ShowDialog" dengan Boolean sebagai tipe datanya. Kemudian dilanjutkan dengan "Book" yang menggunakan Class Book sebagai objectnya.

Lalu kita buat sebuah parameter "CloseEventCallback". EventCallback di Blazor adalah mekanisme untuk menangani event yang terjadi pada komponen. Ini memungkinkan komponen untuk memberi tahu komponen komponen lain ketika suatu peristiwa terjadi. EventCallback sangat berguna untuk komunikasi antar komponen dalam aplikasi Blazor, memungkinkan komponen untuk tetap modular dan terpisah namun tetap dapat berinteraksi satu sama lain. Lalu terdapat dua buah method yaitu "Show" dan "Close". Method Show digunakan untuk mengambil data buku dengan menggunakan "bookid", yang akan dikirimkan dari page/halaman yang lain.

Sedangkan untuk sintaks HTML, pada awal sintaks terdapat sebuah kondisi untuk mengecek nilai dari variable "ShowDialog", apakah true atau false. Jika nilainya alah true, maka akan dilanjutkan dengan mengecek object "Book" apakah null atau tidak. Jika null akan menampilkan pesan "Data tidak tersedia". Jika tidak null atau memiliki data, maka akan menampilkan dialog component. Dapat kita lihat pada sintaks html diatas, untuk data buku akan ditampilkan pada class "Modal-body" (sintaks detail dapat dilihat pada project lampiran).

Sedangkan untuk menutup dialog, selain menggunakan button “x” kita dapat menggunakan custom button atau component yang dapat kita buat sendiri, seperti contoh dibawah ini.

```
<a class="btn btn-outline-primary edit-btn" @onclick="@Close">Close</a>
```

Disini kita menggunakan attribute “onclick” yang memanggil method “Close” dari sintaks C# yang telah kita buat sebelumnya. Method tersebut hanya akan mengganti nilai dari “ShowDialog” menjadi false.

Lalu buka file BookList.razor lalu tambahkan atau ganti beberapa sintaks seperti dibawah ini. Untuk menggunakan component dialog yang telah kita buat diatas, ketikkan sintaks seperti dibawah ini.

Dapat kita lihat “BookDetailDialog.razor” yang telah dibuat sebelumnya, digunakan sebagai komponen pada page ini. Perhatikan attribute dan parameter yang terdapat pada komponen ini. Terdapat parameter CloseEventCallBack dimana parameter tersebut menggunakan method AddEditBookDialog_OnDialogClose yang akan kita buat pada back-end.

```
<BookDetailDialog @ref="BookDetailDialog"
CloseEventCallback="@AddEditBookDialog_OnDialogClose"></BookDetailDialog>
```

Lalu ganti sintak untuk menampilkan detail data buku seperti berikut.

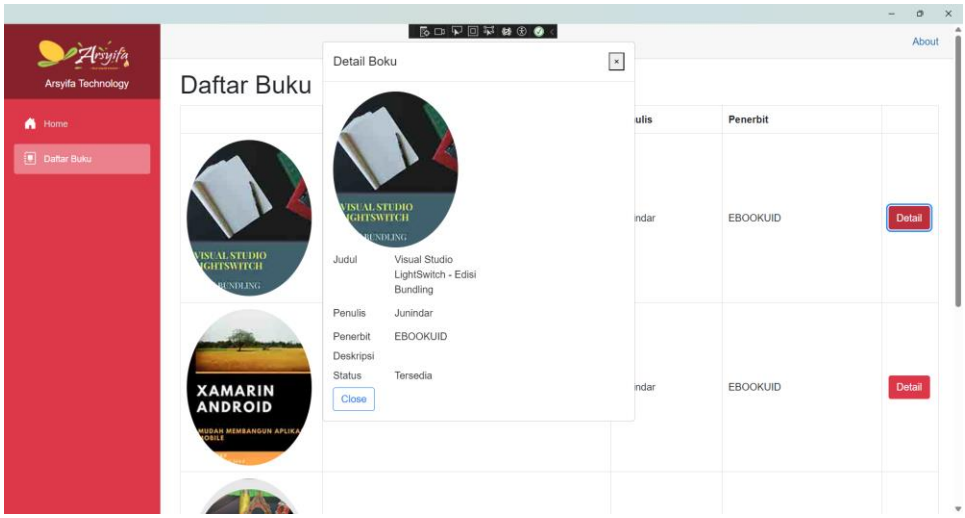
```
<button @onclick="@(() => AddEditBookShow(book.Id))"
class="btn btn-danger table-btn">Detail</button>
```

Terakhir tambahkan sintaks C# dibawah pada back-end.

```
protected BookDetailDialog BookDetailDialog { get; set; }
public void AddEditBookShow(int bookid)
{
    BookDetailDialog.Show(bookid);
}
public async void AddEditBookDialog_OnDialogClose()
{
    StateHasChanged();
}
```

Method “AddEditBookDialog_OnDialogClose“ digunakan untuk melakukan refresh page pada saat dialog ditutup, Sedangkan untuk method “AddEditBookShow“ digunakan memanggil method Show pada BookDetailDialog.razor dengan mengirimkan nilai dari “bookid“, sehingga dialog component akan tampil pada layar.

Jalankan program pada windows machine dan android emulator, dan pastikan mendapatkan hasil seperti dibawah.



Tampilan pada Windows machine

Tampilan pada Mobile

Pada latihan sebelumnya, sintaks HTML dan C# (Code Block) digabung menjadi satu file (*.razor), 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.

- Selanjutnya pada folder Pages tambahkan sebuah class dengan nama "BookList2Base". Seperti yang telah dijelaskan diatas, code behind dari razor komponen ini menggunakan (inherits) dari ComponentBase.

  ```
  public class BookList2Base : ComponentBase
  ```

  dimana sebelumnya kita harus import pada class ini sebuah namespace "`using Microsoft.AspNetCore.Components;`"

- Lalu tambahkan sebuah razor component dengan nama BookList2.razor.

- ```
  @page "/BookList2"
  @inherits BookList2Base
  ```

  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 (BookList2Base). Dan hapus code block pada file ini.

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

  ```
  [Inject]
  public IBookService BookService { get; set; }
  public IEnumerable<Book?>? Books { get; set; }

  protected override async  Task OnInitializedAsync()
  {
      Books =  await BookService.GetBooksAsync();
      await base.OnInitializedAsync();
  }
  ```

  Disini kita melakukan overide pada methos OnInitializeAsync, dapat dilihat pada method tersebut kita panggil method "GetBooksAsync" dari BookService. Dimana hasilnya akan disimpan pada "Books" sebuah public property dengan tipenya adalah "IEnumerable<Book>".
  ```

- Buka kembali file BookList2.razor dan ketikkan sintaks dibawah.

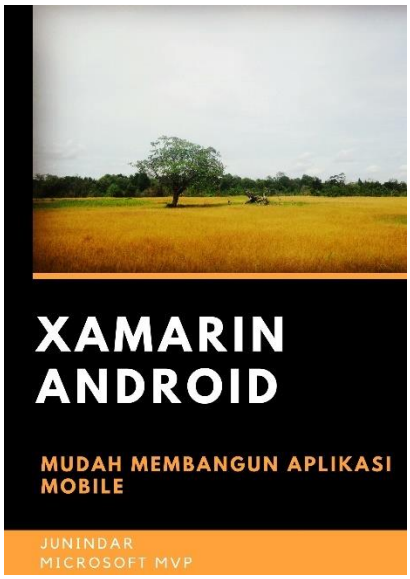
```
@if (Books == null)
{
    <p><em>Data tidak tersedia</em></p>
}
else
{
 //Detail Sintaks pada lampiran
}
```

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. Dan terakhir buka file "NavMenu.razor" lalu ganti href pada "Daftar Buku" menjadi "BookList2". Jalankan program dan 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/dialog-component-pada-blazor-hybrid.html

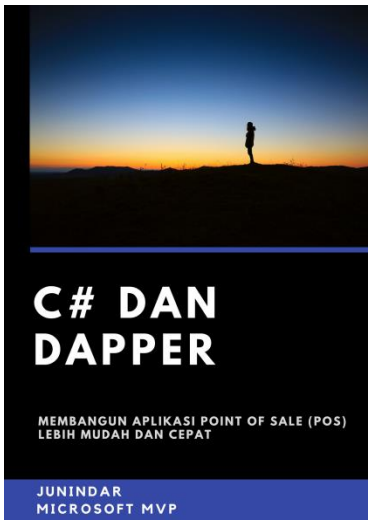
Referensi



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

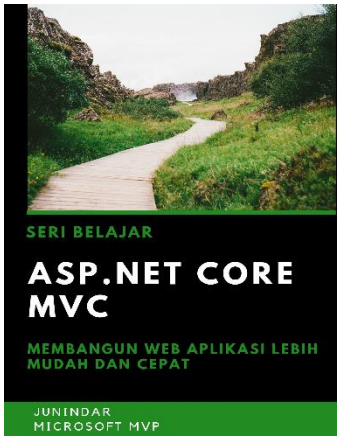
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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".**