IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Membuat Dialog Component

Pada 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 .

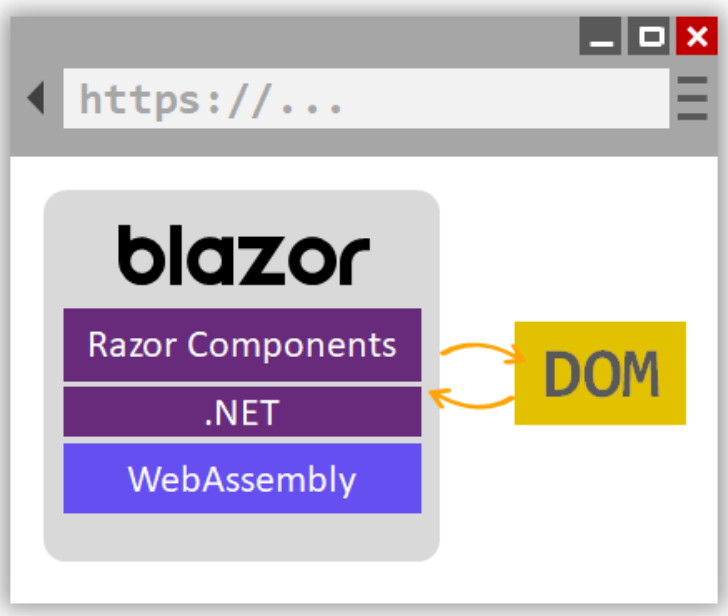
Code .Net berjalan pada WebAssembly, yang artinya kita dapat menjalankan “NET“ didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.

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 .

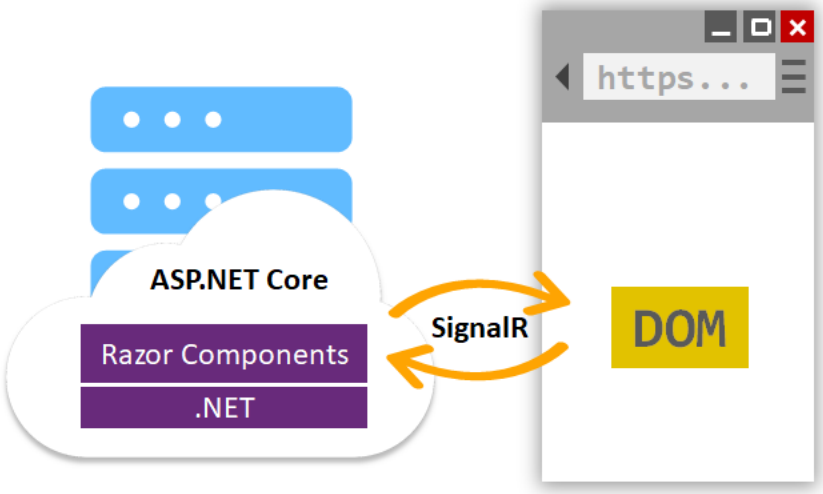
Code .Net berjalan pada Web Assembly, yang artinya kita dapat menjalankan "NET" didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.

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.



Disarankan untuk membaca dan menyeselesaikan latihan pada artikel sebelumnya http://junindar.blogspot.com/2020/02/pengenalan-blazor.html dan http://junindar.blogspot.com/2020/02/create-read-update-dan-delete-crud-pada.html.

Pada artikel ini tidak menjelaskan apa itu blazor, bagaimana bekerja dengan EF Core membuat fungsi CRUD 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 yang digunakan untuk proses CRUD. Untuk memulai latihan pada artikel ini buat terlebih dahulu project Blazor Server side pada Visual Studio 2019.

Note : Link untuk Project Lampiran ada pada Penutup.

Apa kelebihan dari blazor component ini? Kelebihan nya adalah fleksible, ringan dan dapat digunakan lebih dari satu tempat. Pada dasarnya setiap pada/halaman pada blazor adalah sebuah komponen, karena component merupakan elemen dasar dari blazor. Dengan menggunakan kombinasi dari Razor, HTML dan C# maka kita dapat membuat blazor component.

Dibawah ini adalah contoh dari blazor component, terdapat HTML dan C# pada contoh dibawah.

- Component Class

```
@page "/counter"

<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    int currentCount = 0;

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

Pada block function (@code) kita bisa mendifinisikan property maupun method yang digunakan pada HTML. Seperti contoh method "IncrementCount" digunakan pada event onclick pada button "Click Me". Pada saat blazor di-*compile*, maka HTML dan C# akan dikonversi menjadi component class.

- Component Members

  Merupakan member dari component class yang terdapat pada @function block. Kita dapat memiliki lebih dari satu @function block untuk setiap component. Untuk menggunakan component member seperti function maupun variable pada HTML, maka kita harus menggunakan simbol "@" pada awal nama member.

Pada contoh sebelumnya sintaks HTML dan C# kita gabungkan dalam satu file. Sebenarnya kita dapat memisahkan HTML dan C# pada file yang berbeda.

```
using Microsoft.AspNetCore.Components;

namespace LatihanBlazor.Pages
{
    public class CounterClass : ComponentBase
    {
        public int CurrentCount { get; set; }

        public void IncrementCount()
        {
            CurrentCount += 5;
        }
    }
}
```

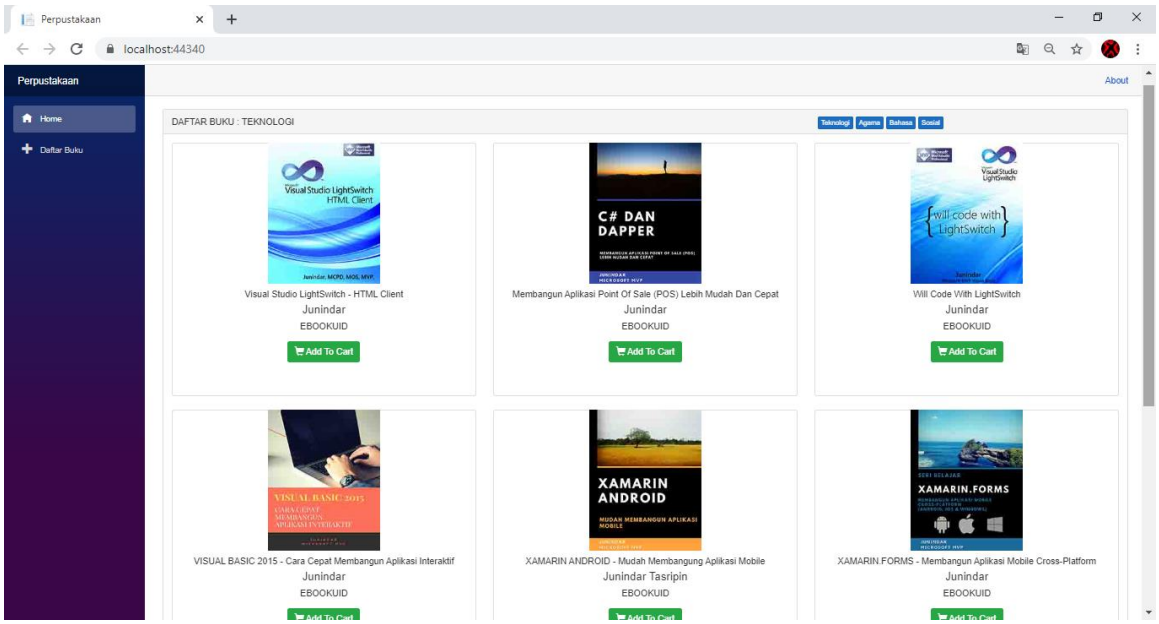
Diatas merupakan *behind class* dari component counter yang telah kita buat sebelumnya. Kita pindahkan sintaks C# pada file terpisah dengan nama CounterClass.cs. Selanjutnya pada razor, kita tambah kan sebuah baris code @inherits CounterClass, seperti pada contoh dibawah ini.

```
@page "/counter"
@inherits CounterClass
<h1>Counter</h1>

<p>Current count: @CurrentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

Setelah selesai dengan penjelasan diatas, maka selanjutnya kita akan membuat blazor component untuk menampilkan data buku, seperti pada gambar dibawah.



Untuk membuat component seperti di atas, Ikuti langkah-langkah dibawah ini.

- Tambah sebuah folder dan ganti namanya menjadi “Components“.
- Lalu tambahkan sebuah method pada IBookRepository dan BookRepository dengan nama “`GetRandomBooks`“ dengan detail sintaks seperti dibawah.

```
public async Task<IEnumerable<Book>> GetRandomBooks(string cat)
{
      return await _dbContext.Books.Where(c=>c.Category.NamaCategory==cat)
                .OrderBy(r => Guid.NewGuid()).Include(b =>
                b.Category).ToListAsync();
}
```

  Method ini berfungsi untuk menampilkan data buku berdasarkan NamaCategory secara random.

- Pada folder Components, tambahkan sebuah class dengan nama CardBookBase.

```
public class CardBookBase : ComponentBase
{

}
```

Pada class diatas akan menggunakan base class dari component. Dibawah ini merupakan detail sintaks pada class tersebut.

```
public string PanelText { get; set; }
[Parameter]
public string CategoryName { get; set; }

[Inject]
public IBookRepository BookRepository { get; set; }
public IEnumerable<Book> Books { get; set; }

protected override async Task OnInitializedAsync()
{
     PanelText = "DAFTAR BUKU : TEKNOLOGI";
     Books = (await BookRepository.GetRandomBooks(CategoryName)).ToList();
}

public async void ShowBookCategoryAsync(string cat)
{
   PanelText = $"DAFTAR BUKU : {cat.ToUpper()}";
   Books = (await BookRepository.GetRandomBooks(cat)).ToList();
   StateHasChanged();
}
```

Property PanelText digunakan untuk menampilkan teks (infomarsi nama category yang dipilih) pada panel, sedangkan CategoryName merupakan parameter yang nantinya digunakan untuk proses pencarian berdasarkan Nama Category. Lalu ada property Books yang merupakan propery untuk menampung hasil dari pencarian data buku berdasarkan CategoryName.

OnInitializedAsync dipanggil ketika component diinisialisai setelah menerima initial parameter dari parent component-nya. Pada method ShowBookCategoryAsync terdapat method “StateHasChanged“, method ini berfungsi untuk memberitahu component bahwa kondisinya telah berubah, sehingga component akan di-*render* ulang.

- Tambahkan Razor Component dengan nama “CardBook.razor“ pada folder Components.

```
@inherits CardBookBase
```

Pada awal baris ketikkan sintaks diatas, untuk memastikan file ini akan terhubung dengan class yang telah kita buat sebelumnya (CardBookBase). Untuk detail sintaks pada file ini dapat dilihat pada project lampiran. Sebagai informasi data-data buku tersebut akan ditampilkan didalam sebuah panel, dimana header panel berisikan teks dan button-button dari Nama Category. Button-button ini berfungsi untuk menampilkan data buku berdasarkan Nama Category.

```
<div class="panel-heading" style="height: 43.38px">
     <label >@PanelText</label>
     <div style="display:inline-block; position:relative; right:-55%;">
          <button type="button" class="btn btn-primary btn-xs"
          @onclick="@(() => ShowBookCategoryAsync("Teknologi"))">Teknologi</button>
          <button type="button" class="btn btn-primary btn-xs"
          @onclick="@(() => ShowBookCategoryAsync("Agama"))">Agama</button>
          <button type="button" class="btn btn-primary btn-xs"
          @onclick="@(() => ShowBookCategoryAsync("Bahasa"))">Bahasa</button>
          <button type="button" class="btn btn-primary btn-xs"
          @onclick="@(() => ShowBookCategoryAsync("Sosial"))">Sosial</button>
      </div>
</div>
```

Pada sintaks di atas dapat dilihat, setiap event click pada button menggunakan method "ShowBookCategoryAsync" dengan mengirimkan nilai ke paramater berupa Nama Category yang akan digunakan untuk pencarian data.

Hasil dari method diatas (ShowBookCategoryAsync) akan ditampilkan pada panel-body, seperti dibawah.

```
@foreach (var book in Books)
{
    <div class="col-md-4">
         <div class="card" style="height: 450px;" align="center">
              <a asp-action="DetailBook" asp-route-id="@book.BookID">
                 <img src="/images/@book.Gambar" height="250" width="200" />
              </a>
              <div style="padding: 9px;color:#333">
                   <h6 style="text-align: center">@book.Judul</h6>
                   <h5 style="text-align: center">@book.Penulis</h5>
                   <h6 style="text-align: center">@book.Penerbit</h6>
                @{
                if (book.Status)
                {
                   <p style="text-align: center">
                      <a class="btn btn-success" role="button"
                         data-bukuId="@book.BookID"
                         data-bukuImage="@book.Gambar"
                         data-bukuJudul="@book.Judul"
                         data-bukuPenulis="@book.Penulis"
                         style="margin-top: 10px;color:white!important">
                      <span class="glyphicon glyphicon-shopping-cart"></span> Add To Cart
                      </a>
                   </p>

                }
                }
              </div>
         </div>
  </div>
}
```

Data-data buku yang ditampilkan akan berbentuk Thumbnail, dan akan menampilkan Judul, Penulis dan Penerbit Buku. Dan jika Status dari buku adalah true atau tersedia maka akan menampilkan button "Add To Cart".

- Selanjutnya untuk menggunakan component yang telah kita buat diatas, buka file index.razor pada folder Pages dan tambah sintaks seperti dibawah.

```
<div class="row">
    <CardBook CategoryName="Teknologi"></CardBook>
</div>
```

Tag CarBook adalah nama dari component, lalu diikuti dengan CategoryName yang merupakan property pada CardBookBase yang telah kita set menjadi parameter. Jalankan project dan pastikan hasilnya sama seperti contoh diatas.

Pada artikel sebelumnya telah dijelaskan bagaimana membuat aplikasi CRUD menggunakan Blazor, pada artikel ini kita akan menggunakan beberapa sintaks dari artikel sebelumnya, jadi pastikan sudah menyelesaikan latihan-latihan pada artikel tersebut. Sekarang kita lanjutkan latihan membuat dialog component untuk proses CRUD, ikuti langkah-langkah dibawah ini.

- Tambahkan sebuah class pada folder Components dengan nama "AddEditBookDialogBase". Lalu copy sintaks pada class BookEditBase.cs yang telah kita buat pada artikel sebelumnya. Dan hapus sintaks berikut.

```
[Inject]
public NavigationManager NavigationManager { get; set; }

[Parameter]
public string BookId { get; set; }
```

Lalu diikuti dengan menghapus method OnInitializedAsync dan NavigateToList.

- Setelah menghapus beberapa baris code seperti yang kita lakukan diatas, maka selanjutnya adalah menambahkan beberapa sintaks yang diperlukan, seperti berikut.

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

"ShowDialog" digunakan sebagai indikator untuk menampilkan atau menghilangkan component pada layar.

- Tambahkan dua buah method Show dan Close seperti dibawah. Method Show digunakan untuk menampilkan data berdasarkan Buku ID dan mengganti property ShowDialog menjadi true. Sedangkan method Close untuk menutup dialog component pada layar.

```
public async void Show(int bookid)
{
     CategoryId = "";
     Categories = await CategoryRepository.GetAll();

     ImageData = string.Empty;
     fs = null;
     file = null;
     if (bookid == 0)
     {
          Book = new Book();
     }
     else
     {

          Book = await BookRepository.GetBookById(bookid);
          CategoryId = Book.CategoryID.ToString();
          ImageData = $"images/{Book.Gambar}";
     }

     ShowDialog = true;
}

public void Close()
{
      ShowDialog = false;
}
```

- Pada method “HandleValidSubmit“ yang telah kita buat sebelumnya terdapat sintaks NavigationManager, hapus baris sintaks tersebut. Dan tambah dua baris sintaks berikut ini.

```
ShowDialog = false;
await CloseEventCallback.InvokeAsync(true);
```

  Lalu lakukan hal yang sama pada method DeleteBook

- Tambah sebuah Razor Component dengan nama “AddEditBookDialog“ pada folder Component. Detail sintaks dapat dilihat pada file lampiran. Sebenarnya sintaks pada file ini sama dengan sintaks BookEdit.razor yang terdapat pada artikel sebelumnya, hanya ada beberapa bagian yang perlu ditambah dan diganti.

  Langkah pertama adalah dengan membuat Modal Page untuk menampilkan data buku, seperti yang ada pada sintaks dibawah. Pada event click button Close, kita gunakan method Close yang terdapat pada AddEditBookDialogBase Class. Sedangkan untuk data-data buku akan ditampilkan pada section body( ). Detail sintaks dapat dilihat pada project lampiran atau dapat menggunakan sintaks pada artikel sebelumnya ()

```
<div class="modal fade show d-block" id="exampleModal" tabindex="-1" role="dialog">
        <div class="modal-dialog" role="document">
            <div class="modal-content" style="width: 800px !important;">
                <div class="modal-header">
                    <h5 class="modal-title" id="titleLabel">Add/Edit/Delete Book</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">

                 @*sintaks Edit Form*@

                </div>
            </div>
        </div>
    </div>
```

- Setelah selesai dengan langkah-langkah diatas, kita lanjutkan dengan mengganti sintaks pada BookListBase.cs. Tambahkan sintaks dibawah ini pada file tersebut.

```
protected AddEditBookDialog AddEditBookDialog { get; set; }

public async void AddEditBookDialog_OnDialogClose()
{
      Books = (await BookRepository.GetAllBooks()).ToList();
      StateHasChanged();
}

public void AddEditBookShow(int bookid)
{
      AddEditBookDialog.Show(bookid);
}
```

Method “AddEditBookDialog_OnDialogClose“ digunakan untuk refresh pada pada saat dialog ditutup, yaitu dengan memanggil method GetAllBooks pada BookRepository class. Sedangkan untuk method “AddEditBookShow“ digunakan memanggil method Show pada class AddEditBookDialog, sehingga dialog component akan tampil pada layar.

Selanjutnya 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.

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

Terdapat parameter CloseEventCallBack dimana parameter tersebut menggunakan method AddEditBookDialog_OnDialogClose yang telah dijelaskan diatas.

Lalu kita lanjutkan dengan mengganti sintaks untuk button “Tambah Buku“, untuk button ini parameter BookID pada method AddEditBookShow kita isi dengan 0,

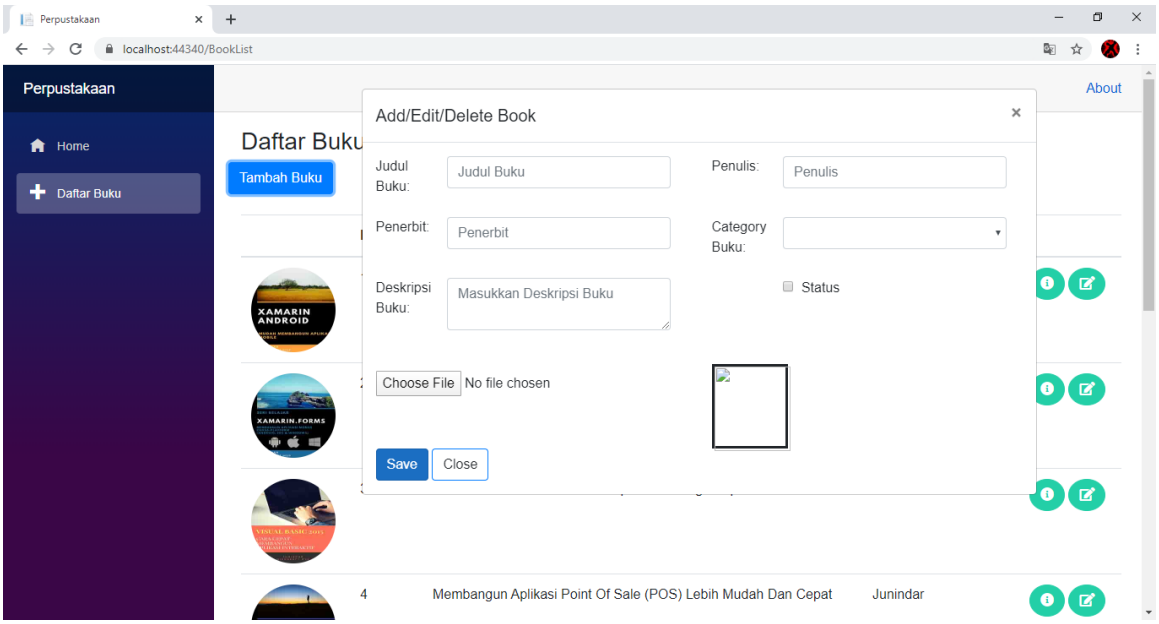
sehingga pada saat component dibuka tidak perlu melakukan pencarian data buku terlebih dahulu.

```
<div class="row">
        <button @onclick="@(() => AddEditBookShow(0))"
        class="btn btn-outline-primary">Tambah Buku</button>
</div>
```

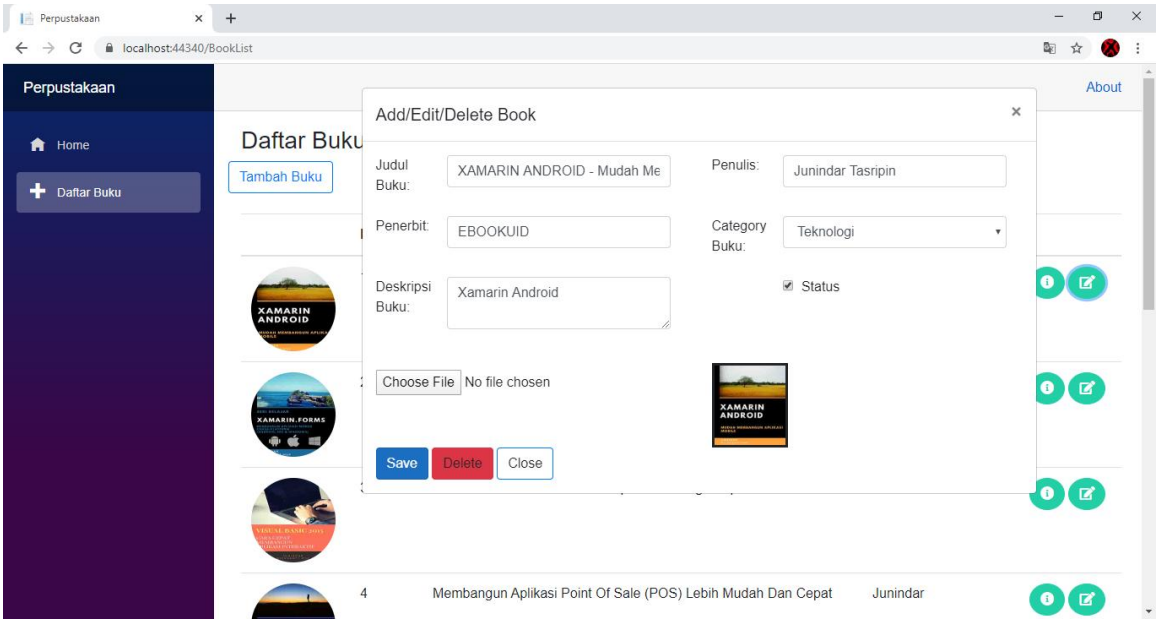
Untuk button Edit, parameter BookID kita gunakan BookId hasil dari query data pada table.

```
<button @onclick="@(() => AddEditBookShow(book.BookID))"
class="btn btn-primary table-btn">  <i class="fas fa-edit"></i>
</button>
```

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



Dialog Tambah Data Buku.



Dialog Ubah Data Buku.

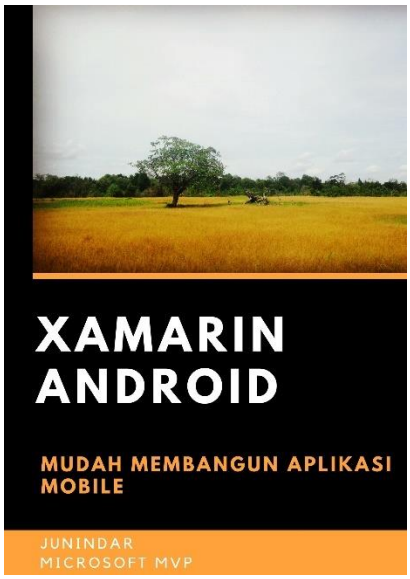
Penutup

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/2020/07/membuat-dialog-component-pada-blazor.html

Semoga bermanfaat.

Wassalam.

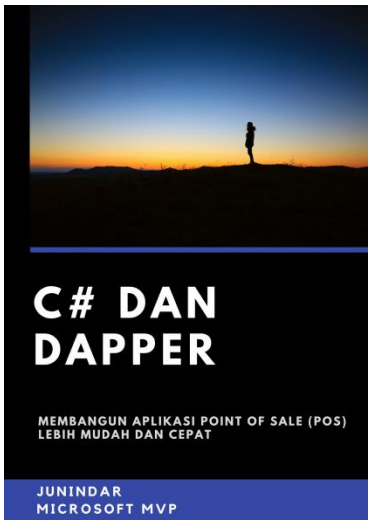
Referensi



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

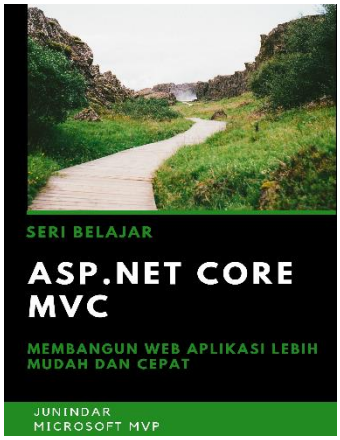
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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".**