IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

MudBlazor Table Pada Blazor – Part 3

Junindar, ST, MCPD, MOS, MCT, MVP

junindar@gmail.com

http://junindar.blogspot.com

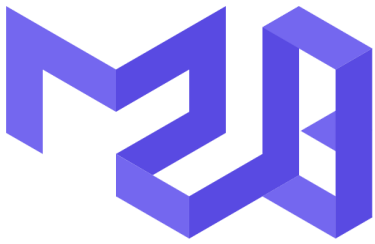
Abstrak

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

Artikel ini melanjutkan dari artikel sebelumnya, yang membahas fungsi-fungsi yang ada pada MudBlazor Table (MudTable). Oleh karena itu sebelum memulai latihan-latihan pada artikel ini, pastikan telah menyelesaikan latihan-latihan pada artikel sebelumnya yang dapat dilihat disini (http://junindar.blogspot.com/2021/12/mudblazor-component-pada-blazor-part-1.html) dan (http://junindar.blogspot.com/2022/07/mudblazor-table-pada-blazor-part-2.html). Perlu diketahui, untuk artikel ini masih akan menggunakan blazor project yang telah kita buat pada artikel sebelumnya.

- Server Side Filtering, Sorting and Pagination

  Pada dua artikel sebelumnya telah dibahas bagaimana cara menampilkan pada MudTable berikut dengan fungsi filter, sort maupun pagination, tetapi semua fitur tersebut menggunakan method Client Side. Dimana seluruh data akan diload terlebih dahulu, sehingga jika kita memiliki data yang sangat besar, maka proses loading data akan sangat lama sekali.

  Sedangkan untuk artikel ini kita akan menggunakan method Server Side, sehingga proses filter, sorting dan paging akan dilakukan pada backend. Dengan menggunakan method ini kita tidak perlu lagi menggunakan property “Items“ dan “Filter“ pada MudTable.

  Tambahkan sebuah Razor Component pada project lalu lalu ikuti langkah-langkah dibawah ini.

  Deklarasikan beberapa variable seperti dibawah ini.

```
private IEnumerable<Book> pagedData;
private MudTable<Book> table;
private int totalItems;
private string searchString = "";
private int[] pageSizeOptions = new int[] {5, 10, 15};
[Inject]
public IBookRepository BookRepository { get; set; }
```

  Selanjutnya kita buat sebuah method dengan nama “ServerReload” seperti dibawah.

```
private async Task<TableData<Book>> ServerReload(TableState state)
{
  IEnumerable<Book> data = await BookRepository.GetAllBooks();
  await Task.Delay(1000);
  data = data.Where(book =>
  {
     if (string.IsNullOrWhiteSpace(searchString)) return true;
     if (book.Penulis.Contains(searchString, StringComparison.OrdinalIgnoreCase))
         return true;
     if (book.Judul.Contains(searchString, StringComparison.OrdinalIgnoreCase))
         return true;
     return false;
 }).ToArray();
 totalItems = data.Count();
switch (state.SortLabel)
 {
    case "id_field":
         data = data.OrderByDirection(state.SortDirection, o => o.BookID);
         break;
    case "judul_field":
         data = data.OrderByDirection(state.SortDirection, o => o.Judul);
         break;
    case "penulis_field":
         data = data.OrderByDirection(state.SortDirection, o => o.Penulis);
         break;
 }
pagedData = data.Skip(state.Page * state.PageSize).Take(state.PageSize).ToArray();
return new TableData<Book>() {TotalItems = totalItems, Items = pagedData};
}
```

Method ini memiliki sebuah parameter dari class “TableState” yang terdapat pada MudBlazor. Lalu pada method ini kita panggil fungsi “`GetAllBooks`” dari BookRepository. Dan hasilnya akan ditampung pada variable “data”, selanjutnya dari hasil yang didapat akan dilakukan proses pencarian berdasarkan keyword yang dimasukkan. Proses yang menggantikan proses Filter pada client side yang kita lakukan pada latihan-latihan sebelumnya.

Untuk proses sorting kita menggunakan switch-case berdasarkan property “SortLabel” pada TableState.

Lalu tambahkan method “OnSearch” seperti dibawah. Method ini akan digunakan pada control “`MudTextField`” search. Terdapat sebuah paramater dengan tipe string, dimana digunakan untuk mengganti nilai dari variable “`searchString`” yang telah kita buat sebelumnya. lalu masih pada method ini kita akan memanggil method “`ReloadServerData`” yang terdapat pada MudTable.

```
private void OnSearch(string text)
{
   searchString = text;
   table.ReloadServerData();
}
```

Dan terakhir untuk sintak C#, kita override method "OnAfterRenderAsync", disini kita gunakan untuk melakukan pengaturan jumlah baris pada setiap halaman dari table.

```
protected override Task OnAfterRenderAsync(bool firstRender)
{
  table.SetRowsPerPage(5);
  return base.OnAfterRenderAsync(firstRender);
}
```

Sekarang kita lanjutkan bekerja pada razor.

Tambahkan MudTable seperti dibawah ini.

```
<MudTable ServerData="@(new Func<TableState, Task<TableData<Book>>>(ServerReload))"
 Hover="true" Loading="true" @ref="table"    LoadingProgressColor="Color.Info">

</MudTable>
```

Pada sintaks diatas terdapat property "**ServerData**" yang menggunakan fungsi "ServerReload" yang telah kita buat sebelumnya. Table akan menunggu fungsi ini dan memperbarui berdasarkan nilai TableData yang hasilkan.

Selanjutnya tambahkan Section "**ToolBarContent**" pada MudTable, seperti dibawah ini.

```
<ToolBarContent>
        <MudText Typo="Typo.h6">Daftar Buku</MudText>
        <MudSpacer />
        <MudTextField T="string" ValueChanged="@(s=>OnSearch(s))" Placeholder="Search"
         Adornment="Adornment.Start"
         AdornmentIcon="@Icons.Material.Filled.Search" IconSize="Size.Medium"
         Class="mt-0"></MudTextField>
</ToolBarContent>
```

Untuk server side kita gunakan property "**ValueChanged**" untuk melakukan proses pencarian. Dimana kita gunakan method "OnSearch" yang telah kita buat sebelumnya. Hal ini berbeda jika kita menggunakan cara client side, dimana pada client side kita menggunakan property "**@bind-Value**".

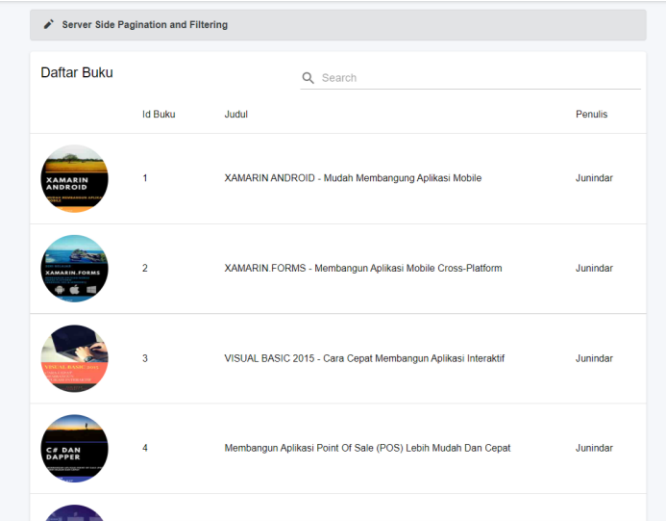
```
<HeaderContent>
   <MudTh></MudTh>
   <MudTh><MudTableSortLabel SortLabel="id_field" T="Book">
         Id Buku</MudTableSortLabel></MudTh>
   <MudTh><MudTableSortLabel SortLabel="judul_field"
         T="Book">Judul</MudTableSortLabel></MudTh>
   <MudTh><MudTableSortLabel SortLabel="penulis_field"
        T="Book">Penulis</MudTableSortLabel></MudTh>
</HeaderContent>
```

Pada sintaks diatas, digunakan untuk membuat header pada table. Dimana yang perlu diperhatikan adalah property "**SortLabel**". Pastikan namanya sesuai dengan yang ada pada method "ServerReload".

```
<RowTemplate>
   <MudTd DataLabel="Gambar">
      <img src="@($"images/{@context.Gambar}")" class="rounded-circle"
       width="100" height="100" />
   </MudTd>
   <MudTd  DataLabel="BookID">@context.BookID</MudTd>
   <MudTd DataLabel="Judul">@context.Judul</MudTd>
   <MudTd DataLabel="Penulis">@context.Penulis</MudTd>
</RowTemplate>

<NoRecordsContent>
    <MudText>No matching records found</MudText>
</NoRecordsContent>
<LoadingContent>
    <MudText>Loading...</MudText>
</LoadingContent>
<PagerContent>
    <MudTablePager PageSizeOptions="pageSizeOptions" />
</PagerContent>
```

Dan terakhir adalah dengan membuat sintaks untuk "`RowTemplate`", "`NoRecordsContent`", "`LoadingContent`" dan "`PagerContent`" seperti pada sintaks diatas. Lalu jalankan program dan pastikan seluruh fungsi (search, paging dan sorting) berjalan dengan baik.



- Inline Edit Mode

  Setelah selesai dengan latihan diatas, kita lanjutkan dengan membuat Inline Edit Mode pada table. Pada MudTable terdapat fungsi dimana kita dapat melakukan mengganti data langsung pada baris yang dipilih. MudTable menyediakan Input element sehingga kita dapat langsung mengganti value pada baris yang dipilih.

  Tambahkan sebuah razor component dan copykan sintaks pada latihan sebelumnya kedalam file ini. Lalu ikuti langkah-langkah dibawah ini.

Tambahkan sebuah variable seperti berikut.

```
private Book elementBeforeEdit;
```

Variable ini akan digunakan untuk menyimpan data-data pada baris yang dipilih, sebelum dilakukan proses edit.

```
private void BackupItem(object book)
{
    elementBeforeEdit = new Book()
    {
            BookID = ((Book)book).BookID,
            Judul = ((Book)book).Judul,
            Penulis = ((Book)book).Penulis
    };
}
```

Lalu untuk untuk menyimpan data-data pada baris yang dipilih, sebelum dilakukan proses edit kita buat sebuah method seperti diatas. Selanjutnya buat sebuah method untuk menyimpan hasil edit ke database, seperti pada sintaks dibawah ini.

```
private async void SaveItem(object book)
{
   var updateBook = await BookRepository.GetBookById(((Book) book).BookID);
   updateBook.Judul=((Book)book).Judul;
   updateBook.Penulis=((Book)book).Penulis;
   await  BookRepository.Update(updateBook);
}
```

Note : untuk detail sintaks update kedatabase dapat dilihat pada project lampiran.

Dan terakhir untuk sintak C#, buat sebuah method yang berfungsi untuk melakukan reset data ke original data. Fungsi ini digunakan jika user melakukan cancel edit pada baris yang dipilih.

```
private void ResetItemToOriginalValues(object book)
{
   ((Book)book).BookID = elementBeforeEdit.BookID;
   ((Book)book).Judul = elementBeforeEdit.Judul;
   ((Book)book).Penulis = elementBeforeEdit.Penulis;
}
```

Setelah selesai dengan C#, kita lanjutkan bekerja pada Razor.

```
<MudTable ServerData="@(new Func<TableState, Task<TableData<Book>>>(ServerReload))" Hover="true"
          CanCancelEdit="true" ReadOnly="false"
          Loading="false" @ref="table" CommitEditTooltip="Commit Edit"
          RowEditPreview="BackupItem" RowEditCancel="ResetItemToOriginalValues"
          RowEditCommit="SaveItem"
          LoadingProgressColor="Color.Info">
```

Ganti atau tambahkan property pada MudTable seperti diatas. Ketiga method yang telah kita buat diatas akan digunakan pada property berikut.

"**RowEditPreview**" menggunakan method "BackupItem", "**RowEditCancel**" menggunakan method "ResetItemToOriginalValues" dan terakhir "**RowEditCommit**"

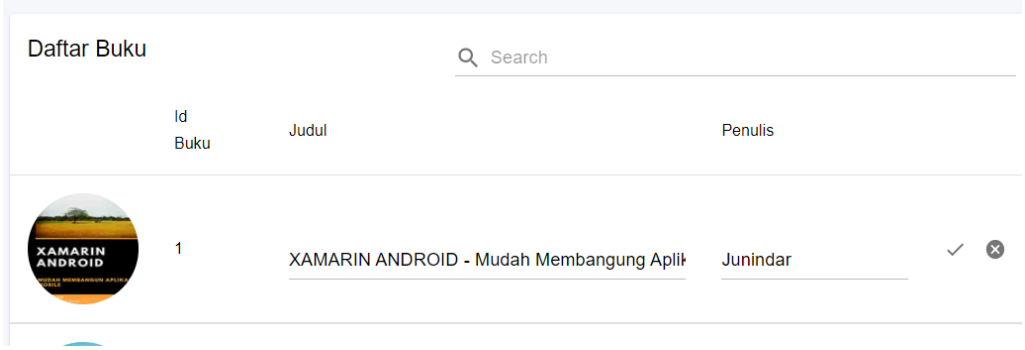
menggunakan method "SaveItem". Dan pastikan property "**CanCancelEdit**" di set menjadi true, dan "**ReadOnly**" menjadi false.

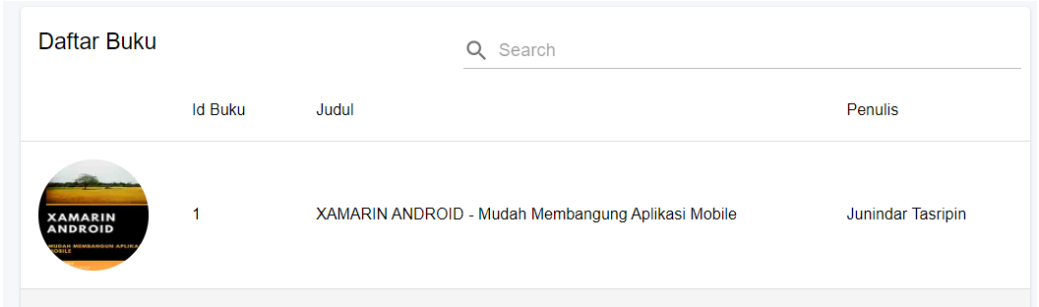
Selanjutnya kita perlu tambahkan sintaks untuk "**RowEditingTemplate**" pada saat baris dipilih, dengan kata lain mengganti control menjadi input element, seperti dibawah ini.

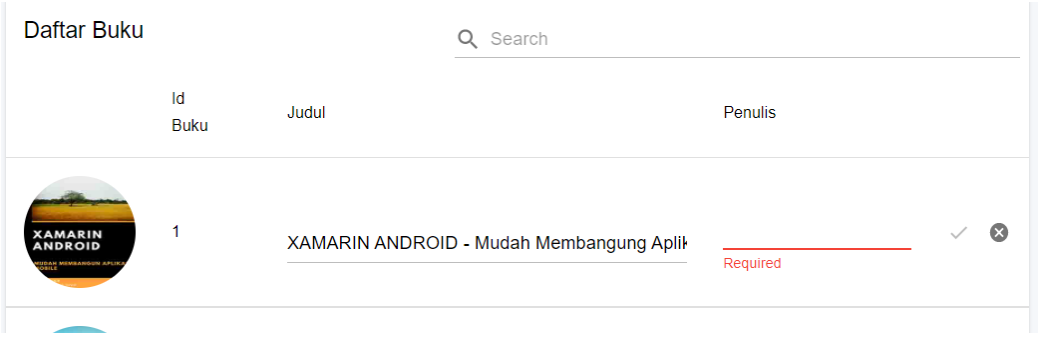
```
<RowEditingTemplate>
        <MudTd DataLabel="Gambar">
            <img src="@($"images/{@context.Gambar}")" class="rounded-circle" width="100"
             height="100" />
        </MudTd>
        <MudTd DataLabel="BookID">@context.BookID</MudTd>
        <MudTd DataLabel="Judul">
            <MudTextField @bind-Value="@context.Judul" Required />
        </MudTd>
        <MudTd DataLabel="Penulis">
            <MudTextField @bind-Value="@context.Penulis" Required />
        </MudTd>
</RowEditingTemplate>
```

Pada sintaks diatas dapat dilihat, hanya Judul dan penulis yang dapat diganti datanya. Dan kedua kolom tersebut required yang artinya data harus diisi.

Jalankan program dan pastikan semua fungsi yang telah dijelaskan diatas berfungsi dengan baik.



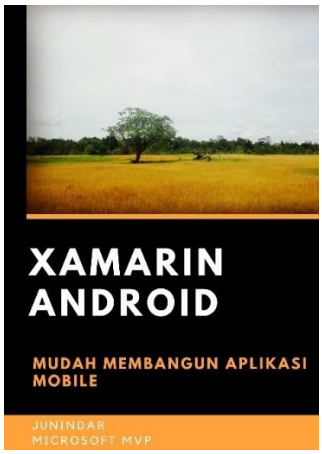



Daftar Buku
Search
Id Buku
Judul
Penulis
1
XAMARIN ANDROID - Mudah Membangung Aplik
Required

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/2022/08/mudblazor-table-pada-blazor-part-3.html

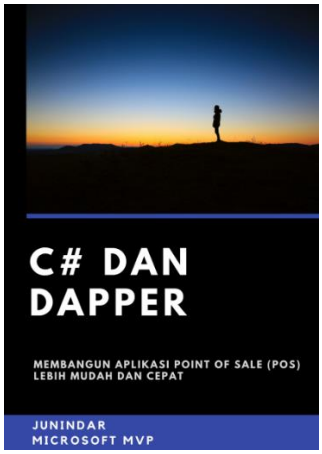
Referensi



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

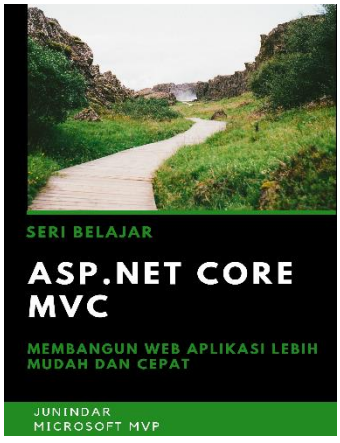
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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

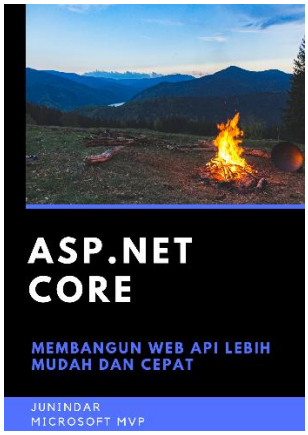


https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

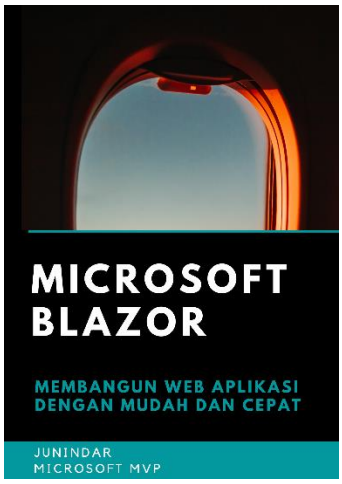
https://play.google.com/store/books/details/Junindar_ASP_NET_MVC_Membangun_Aplikasi_Web_Lebih?id=XLlyDwAAQBAJ



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



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

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".**