MudBlazor Input Component

Pada Blazor – Part 2

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

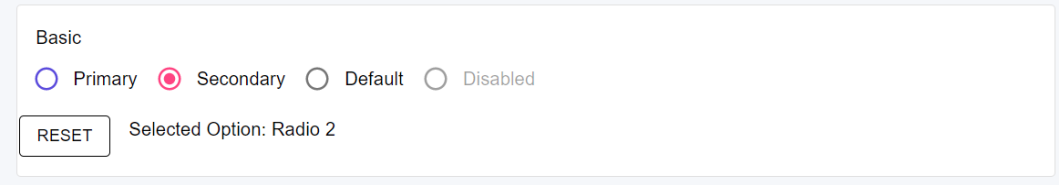
MudBlazor adalah sebuah material design component framework yang dibangun khusus untuk Blazor (https://mudblazor.com/). Terdapat banyak komponen pada MudBlazor seperti chart, grid dan lain-lain untuk membantu dalam membangun aplikasi web dengan menggunakan blazor. Seluruh komponen pada MudBlazor dibangun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

Artikel ini melanjutkan dari artikel sebelumnya, yang membahas komponen-komponen Input yang terdapat pada MudBlazor. Komponen-komponen tersebut sering digunakan dalam sebuah aplikasi web. Beberapa komponen yang akan dijelaskan pada artikel ini adalah Radio Button, Switch, CheckBox dan AutoComplete. Dikarenakan artikel ini masih melanjutkan fitur-fitur yang ada pada MudBlazor, maka sebelum memulai latihan-latihan pada artikel ini, pastikan telah menyelesaikan latihan-latihan pada artikel sebelumnya yang dapat dilihat disini. http://junindar.blogspot.com/2022/10/mudblazor-input-component-pada-blazor.html . Perlu diketahui, untuk artikel ini masih akan menggunakan blazor project yang telah kita buat pada artikel sebelumnya.

- Radio button

  Radio button adalah komponen yang digunakan untuk memungkinkan pengguna dalam memilih satu pilihan dalam sebuah group pilihan. Biasanya digunakan jika jumlah dari pilihan dalam suatu group tidak terlalu banyak.

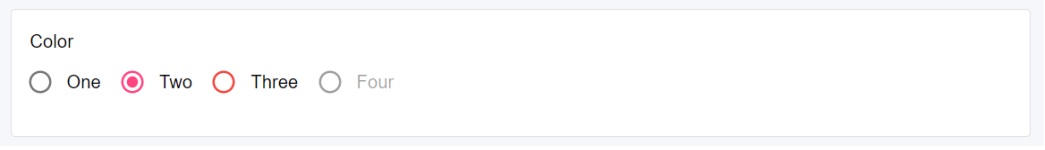
  1. Basic




```
<MudRadioGroup @bind-SelectedOption="@SelectedOption">
   <MudRadio Option="@("Radio 1")" Color="Color.Primary">Primary</MudRadio>
   <MudRadio Option="@("Radio 2")" Color="Color.Secondary">Secondary</MudRadio>
   <MudRadio Option="@("Radio 3")">Default</MudRadio>
   <MudRadio Option="@("Radio 4")" Color="Color.Primary"
   Disabled="true">Disabled</MudRadio>
</MudRadioGroup>
<MudButton Variant="Variant.Outlined" OnClick="Reset">Reset</MudButton>
<MudText Class="ml-4">Selected Option: @SelectedOption</MudText>

@code {
public string SelectedOption { get; set; }
private void Reset()
    {
        SelectedOption = null;
    }
}
```

Pada MudRadio (Radio Button) terdapat atribut “Option” yang digunakan untuk Value pada setiap button. Tipe dari value tersebut harus sama dengan “`@bind-SelectedOption`” pada MudRadioGroup. Pastikan Radio Button berada didalam Komponen MudRadioGroup.

2. Color, Dense dan Size



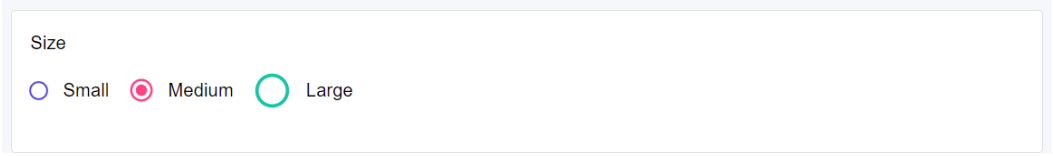

```
<MudRadioGroup T="int">
     <MudRadio Option="1" Color="Color.Primary"
     UnCheckedColor="Color.Default">One</MudRadio>
     <MudRadio Option="2" Color="Color.Secondary"
     UnCheckedColor="Color.Default">Two</MudRadio>
     <MudRadio Option="3" Color="Color.Success"
     UnCheckedColor="Color.Error">Three</MudRadio>
</MudRadioGroup>
```

Untuk mengganti warna dari MudRadio kita dapat menggunakan attribut “Color”, seperti pada contoh diatas. Sedangkan untuk mengganti warna jika Radio Button tidak dipilih maka kita gunakan atribut “`UnCheckedColor`”.

Seperti pada komponen MudBlazor pada umumnya, kita dapat mengatur padding dari komponen dengan menggunakan atribut “Dense” menjadi true, seperti pada sintaks dibawah ini.

```
<MudRadio Option="true" Color="Color.Primary" Dense="true">Dense</MudRadio>
<MudRadio Option="false" Color="Color.Secondary" Dense="false">Normal</MudRadio>
```

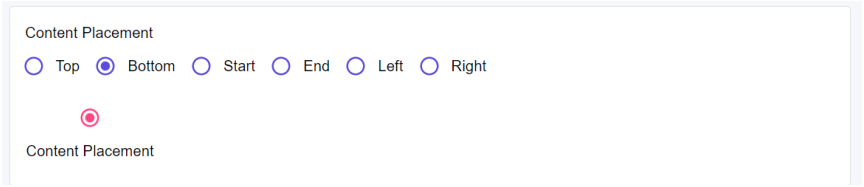
Terdapat 3 buah ukuran pada MudRadio (Small, Medium dan Large). Untuk mengganti dari ukuran Radio Button kita gunakan atribut Size, seperti pada contoh dibawah ini.




```
<MudRadioGroup @bind-SelectedOption="Radio_Size">
   <MudRadio Option="1" Color="Color.Primary" Size="Size.Small">Small</MudRadio>
   <MudRadio Option="2" Color="Color.Secondary"
   Size="Size.Medium">Medium</MudRadio>
   <MudRadio Option="3" Color="Color.Tertiary" Size="Size.Large">Large</MudRadio>
</MudRadioGroup>
```

3. Content Placement

Untuk mengatur posisi dari Child Content pada Radio Button, kita gunakan atribut Placement. Pada MudBlazor terdapat Public Enum Placement yang digunakan untuk mengatur posisi tersebut. Seperti pada contoh dibawah ini.



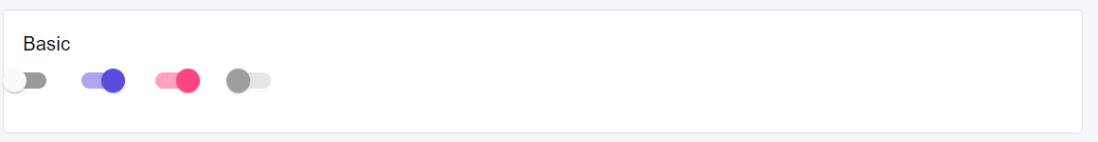

```
<div class="row">
     <MudRadioGroup @bind-SelectedOption="@Placement">
        <MudRadio Color="Color.Primary" Option="@(Placement.Top)">Top</MudRadio>
        <MudRadio Color="Color.Primary"
        Option="@(Placement.Bottom)">Bottom</MudRadio>
        <MudRadio Color="Color.Primary"
        Option="@(Placement.Start)">Start</MudRadio>
        <MudRadio Color="Color.Primary" Option="@(Placement.End)">End</MudRadio>
        <MudRadio Color="Color.Primary"
        Option="@(Placement.Left)">Left</MudRadio>
        <MudRadio Color="Color.Primary"
        Option="@(Placement.Right)">Right</MudRadio>
     </MudRadioGroup>
</div>
```

```
<div class="row">
    <MudRadioGroup T="string">
       <MudRadio T="string" Placement="@Placement"
       Color="Color.Secondary">Content Placement</MudRadio>
    </MudRadioGroup>
</div>
@code {
        public Placement Placement { get; set; } = Placement.Right;
}
```

- Switch

  Switch adalah komponen yang mirip dengan Checkbox dengan visual yang berbeda, untuk Switch menggunakan visual seperti sakelar.

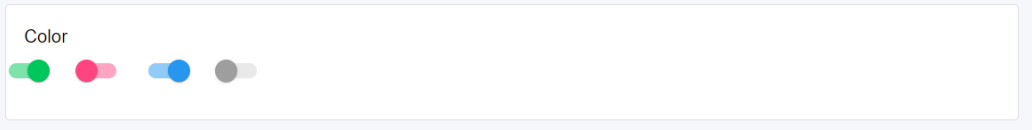
  1. Basic




```
<MudSwitch @bind-Checked="@Basic_Switch1" />
<MudSwitch @bind-Checked="@Basic_Switch2" Color="Color.Primary" />
<MudSwitch @bind-Checked="@Basic_Switch3" Color="Color.Secondary" />
<MudSwitch T="bool" Disabled="true" />
@code {

    public bool Basic_Switch1 { get; set; } = false;
    public bool Basic_Switch2 { get; set; } = true;
    public bool Basic_Switch3 { get; set; } = true;
}
```

  MudSwitch adalah nama komponen ini pada MudBlazor. Switch hanya memiliki dua nilai, seperti true-false, 0-1, yes-no dan lain-lain.

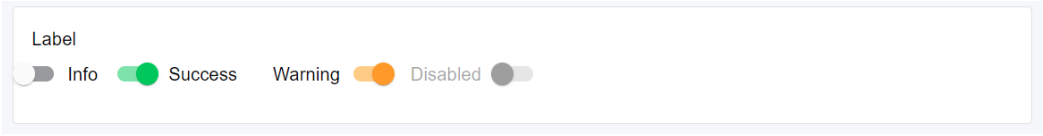
  2. Color dan Label




```
<MudSwitch @bind-Checked="@Basic_Switch1" Color="Color.Success"
UnCheckedColor="Color.Error" />
<MudSwitch @bind-Checked="@Basic_Switch2" Color="Color.Primary"
UnCheckedColor="Color.Secondary" />
<MudSwitch @bind-Checked="@Basic_Switch3" Color="Color.Info"
UnCheckedColor="Color.Warning" />
<MudSwitch T="bool" Disabled="true" UnCheckedColor="Color.Dark" />

@code {

    public bool Basic_Switch1 { get; set; } = false;
    public bool Basic_Switch2 { get; set; } = true;
    public bool Basic_Switch3 { get; set; } = true;
}
```

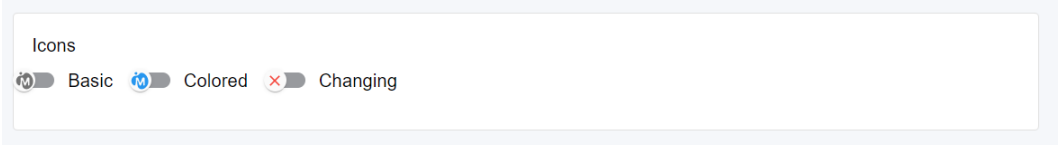
Untuk mengganti warna dari MudSwitch kita dapat menggunakan attribut "Color", seperti pada contoh diatas. Sedangkan untuk mengganti warna jika Switch tidak dipilih maka kita gunakan atribut "`UnCheckedColor`".

Jika ingin menambahkan label pada switch, gunakan atribute "Label" seperti pada contoh dibawah. Dan kita bisa juga mengatur posisi dari label dengan menggunakan atribute LabelPosition.




```
<MudSwitch @bind-Checked="@Label_Switch1" Label="Info" Color="Color.Info" />
<MudSwitch @bind-Checked="@Label_Switch2" Label="Success" Color="Color.Success" />
<MudSwitch @bind-Checked="@Label_Switch3" Label="Warning"
LabelPosition="LabelPosition.Start" Color="Color.Warning" />
<MudSwitch T="bool" Disabled="true" Label="Disabled"
LabelPosition="LabelPosition.Start" />
```

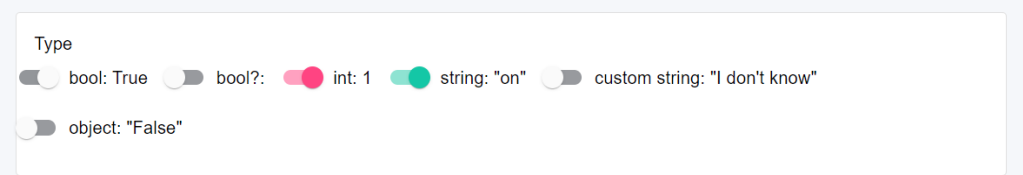
3. Icon




```
<MudSwitch @bind-Checked="@_checked1"
ThumbIcon="@Icons.Custom.Brands.MudBlazor">Basic</MudSwitch>

<MudSwitch @bind-Checked="@_checked2" ThumbIcon="@Icons.Custom.Brands.MudBlazor"
ThumbIconColor="Color.Info">Colored</MudSwitch>

<MudSwitch @bind-Checked="@_checked3" ThumbIcon="@(_checked3==true ?
Icons.Material.Filled.Done : Icons.Material.Filled.Close)"
ThumbIconColor="@(_checked3==true ? Color.Success :
Color.Error)">Changing</MudSwitch>

@code {

    bool _checked1 = false;
    bool _checked2 = false;
    bool _checked3 = false;
}
```

Selain label kita juga dapat menambahkan Icon pada komponen Switch. Gunakan atribute "ThumbIcon", sedangkan untuk icon-nya kita dapat mengunakan class Icon yang terdapat pada MudBlazor seperti pada contoh diatas. Selain itu untuk mengganti warna dari icon kita gunakan atribut "ThumbIconColor". Coba perhatikan pada contoh diatas. Untuk Switch yang ketiga terdapat kondisi dimana jika bernilai true maka icon yang digunakan berbeda dengan jika bernilai false, begitu juga dengan Color.

4. Type



Pada contoh-contoh sebelumnya tipe yang kita gunakan adalah boolean dimana hanya memiliki nilai true dan false. Lalu bagaimana jika kita menggunakan tipe data yang lain seperti integer, string maupun object.

```
<MudSwitch @bind-Checked="integer" Color="Color.Secondary">int:
@integer</MudSwitch>

@code {
    public int integer { get; set; } = 1;
}
```

Untuk contoh diatas, default value dari nilai integer adalah “1” yang artinya true atau *checked*. Jadi pada saat halaman dibuka maka Switch dalam keadaan dipilih (*checked*). Jika user tidak dipilih (*unchecked*) maka nilai integer akan berubah menjadi “0”.

```
<MudSwitch @bind-Checked="str" Color="Color.Tertiary">string: "@(str)"</MudSwitch>

@code {
    public string str { get; set; } = "on";
}
```

Untuk tipe string value yang dikenal adalah “on” (*checked*) dan “off” (*unchecked*).

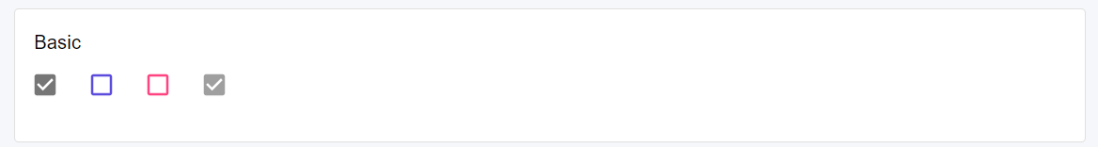
Sedangkan jika ingin menggunakan custom string, kita perlu membuat sebuah class untuk mengganti nilai dari Boolean menjadi string, seperti pada sintaks dibawah ini.

```
<MudSwitch @bind-Checked="customstr" Color="Color.Error" Converter="@(new
CustomStringToBoolConverter())"> custom string: "@(customstr)"</MudSwitch>

@code {
    public string customstr { get; set; } =
    CustomStringToBoolConverter.NullString;

    public class CustomStringToBoolConverter : BoolConverter<string>
    {
        public CustomStringToBoolConverter()
        {
            SetFunc = OnSet;
            GetFunc = OnGet;
        }
```

```
        public const string TrueString = "Yes, please";
        public const string FalseString = "No, thanks";
        public const string NullString = "I don't know";

        private string OnGet(bool? value) => value == null ? NullString :
        (value == true ? TrueString : FalseString);

        private bool? OnSet(string arg)
        {
            try
            {
                if (arg == TrueString)
                    return true;
                if (arg == FalseString)
                    return false;
                return null;
            }
            catch (FormatException e)
            {
                UpdateSetError("Conversion error: " + e.Message);
                return null;
            }
        }
    }
}
```

- Checkbox

  Checkbox merupakan komponen yang digunakan untuk pengguna membuat pilihan yang lebih dari satu dari daftar pilihan yang tersedia. Pada MudBlazor nama dari checkbox ini adalah "`MudCheckBox`"




```
<MudCheckBox @bind-Checked="@Basic_CheckBox1"></MudCheckBox>
<MudCheckBox @bind-Checked="@Basic_CheckBox2" Color="Color.Primary"></MudCheckBox>
<MudCheckBox @bind-Checked="@Basic_CheckBox3"
Color="Color.Secondary"></MudCheckBox>
<MudCheckBox @bind-Checked="@Basic_CheckBox4" Disabled="true"></MudCheckBox>
```

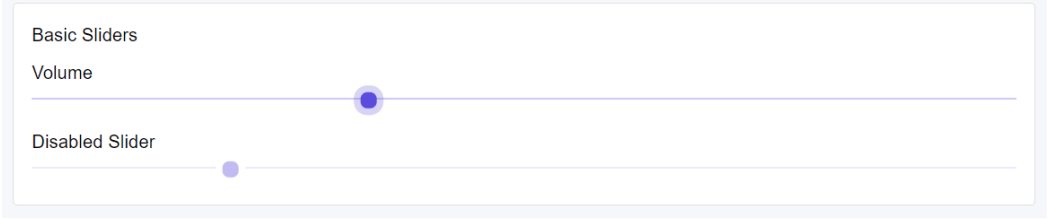
  Untuk fungsi-fungsinya Checkbox memiliki kesamaan dengan Switch, sehingga disini tidak akan dibahas secara detail.

  Note : Untuk Sintaks detail dapat dilihat pada project lampiran.

- Slider

  Slider adalah komponen yang memungkinkan pengguna untuk memilih dari rentang nilai. Untuk mengubah nilai tersebut pengguna hanya perlu menggeser titik atau point kekiri atau kekanan.

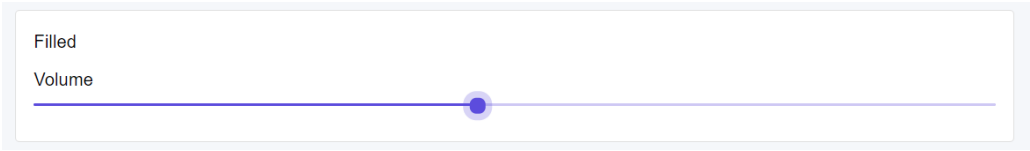
1. Basic




```
<MudSlider Value="@value">Volume</MudSlider>
<MudSlider Disabled="true" Value="@(20)">Disabled Slider</MudSlider>

@code {
        double value = 50.0;
}
```

MudSlider adalah nama dari Slider pada MudBlazor. Seperti pada contoh diatas, "Value" pada slider pertama menggunakan nilai dari *field* "value" pada C#. Untuk slider kedua adalah contoh untuk membuat slider menjadi *disable*.

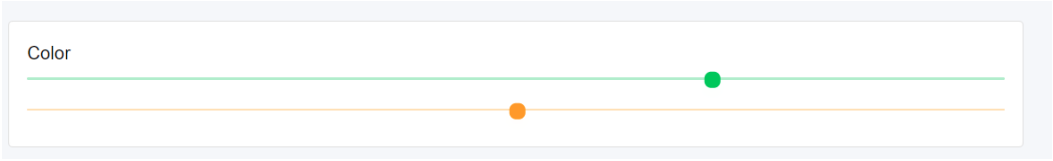
2. Filled




```
<MudSlider Value="@value" Variant="Variant.Filled">Volume Filled</MudSlider>
@code {
        double value = 50.0;
}
```

Slider memiliki fitur Variant.Filled, dimana bagian nilai yang dipilih garisnya menjadi lebih tebal, sehingga dapat membedakan bagian dari nilai yang dipilih maupun tidak.

3. Color, Step dan Min-Max Value

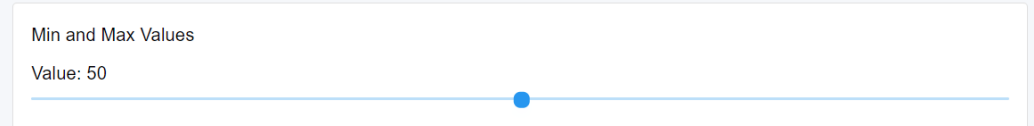


Seperti pada komponen-komponen sebelumnya, slider juga memiliki fitur untuk mengganti warna dengan menggunakan atribut Color.

```
<MudSlider Step="10" Value="70" Color="Color.Success" />
```

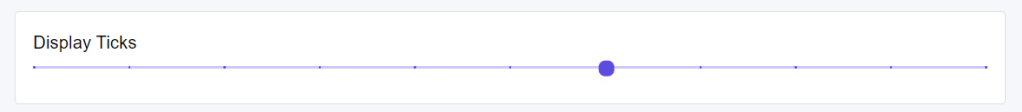
Pada sintaks diatas terdapat atribut "Step", yang digunakan untuk mengatur langkah (step) disetiap perpindahan.

Selain itu kita dapat mengatur nilai Minimal (Min) dan Maximum (Max) pada slider. Sehingga tidak memungkikan pengguna untuk memasukkan data diluar dari range yang telah kita tentukan, seperti pada contoh dibawah ini.



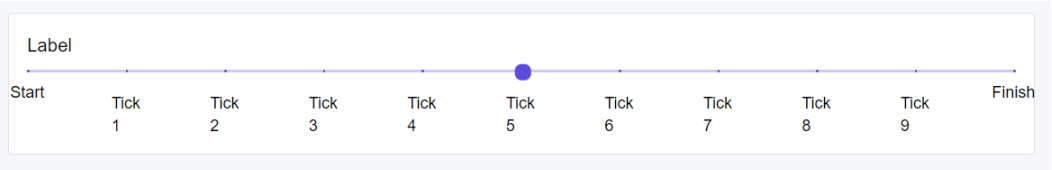

```
<MudSlider @bind-Value="value1" Min="20" Max="80" Color="Color.Info">Value:
@value1.ToString()</MudSlider>
```

4. Display Tick dan Label




```
<MudSlider TickMarks="true" Step="10" Value="@value" />
```

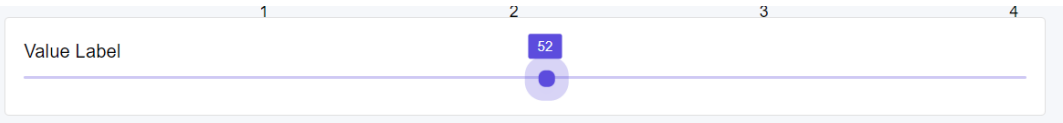
Untuk menampilkan “Tick“ pada slider kita gunakan atribut “`TickMarks`“ menjadi true. Agar jumlah “TickMark“ tidak terlalu banyak sebaiknya gunakan Step sesuai dengan range yang digunakan.




```
<MudSlider TickMarks="true" TickMarkLabels="@labels" Step="10" Value="@valueInt" />
@code {

int valueInt = 50;
string[] labels = new string[] { "Start", "Tick 1", "Tick 2", "Tick 3",
"Tick 4", "Tick 5", "Tick 6", "Tick 7", "Tick 8", "Tick 9", "Finish" };

}
```

Untuk mengatifkan label pada TickMark, kita gunakan atribut “TickMarkLabels”, nilai yang digunakan untuk atribut ini adalah array (string), seperti pada contoh diatas.

5. Value Label dan Size



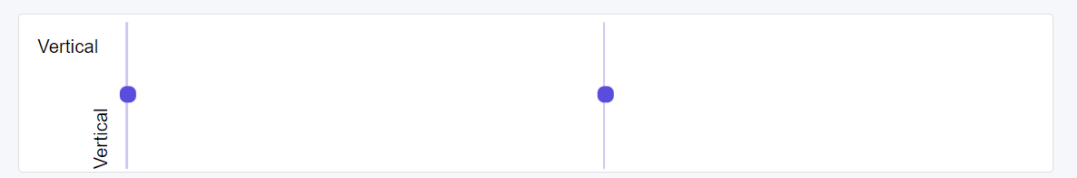
Untuk menampilkan Value pada saat slider digeser seperti pada gambar diatas, kita gunakan atribut “`ValueLabel`“ menjadi true.

Sedangkan untuk mengganti ukuran/size dari slider kita gunakan atribut "`Size`". Terdapat 3 buah ukuran pada slider yaitu small, medium dam large.

```
<MudSlider Size="Size.Small" Value="@valueInt" />
<MudSlider Size="Size.Medium" Value="@valueInt" />
<MudSlider Size="Size.Large" Value="@valueInt" />
```

6. Vertical

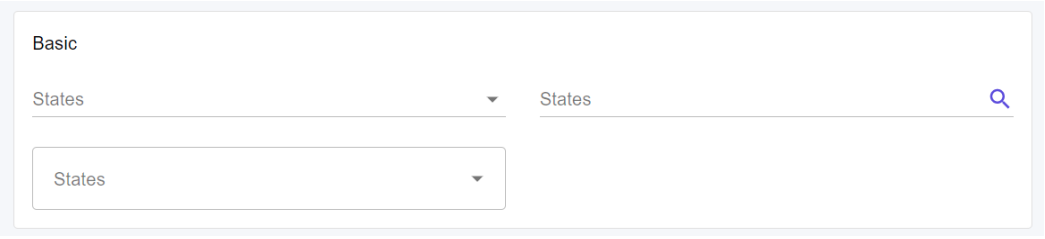
Pada contoh-contoh sebelumnya slider ditampilkan secara horizontal. Tetapi kita juga dapat mengatur agar slider ditampilkan menjadi vertical dengan menggunakan atribut/property "`Vertical`" menjadi true. Sehingga mendapatkan hasil seperti dibawah.



- Autocomplete

Autocomplete adalah komponen yang memiliki daftar pilihan yang banyak. Tidak seperti komponen Select, untuk Autocomplete pengguna tidak perlu mengetahui daftar item secara lengkap. Pengguna hanya perlu memasukkan beberapa karakter maka secara otomatis autocomplete akan memanggil fungsi pencarian dan menampilkan hasil yang sesuai. Fungsi pencarian bahkan dapat berjalan secara *asynchronously* seperti untuk query pada Database.

1. Basic




```
<MudAutocomplete T="string" Label="States" @bind-Value="value1"
SearchFunc="@Search1" ResetValueOnEmptyText="true" />

<MudAutocomplete T="string" Label="States" @bind-Value="value2"
SearchFunc="@Search2" ResetValueOnEmptyText="true"
AdornmentIcon="@Icons.Material.Filled.Search" AdornmentColor="Color.Primary" />

<MudAutocomplete T="string" Label="States" @bind-Value="value3"
SearchFunc="@Search3" Margin="Margin.None"
Variant="Variant.Outlined" ResetValueOnEmptyText="true" />
```

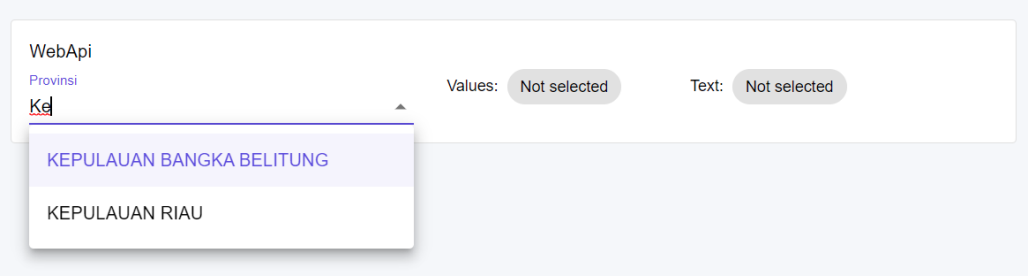
Pada gambar diatas terdapat 3 buah Autocomplete dengan bentuk yang berbeda-beda. Yang pertama merupakan basic Autocomplete, untuk yang kedua kita tambahkan Icon (`AdornmentIcon`) dan Color (`AdornmentColor`). Sedankan yang terakhir kita gunakan atribut "`Variant`" (Outlined). Ketiga Autocomplete diatas memiliki atribut "`ResetValueOnEmptyText`" dan "`SearchFunc`". ResetValueOnEmptyText digunakan untuk me-reset value jika pengguna menghapus text pada autocomplete. Sedangkan SearchFunc berfungsi untuk mengembalikan daftar nilai dari hasil pencarian. Oleh karena itu kita perlu membuat method untuk pencarian yang diperlukan tersebut, seperti contoh dibawah.

```
private async Task<IEnumerable<string>> Search1(string value)
{
    await Task.Delay(5);
    if (string.IsNullOrEmpty(value))
       return states;
    return states.Where(x => x.Contains(value,
    StringComparison.InvariantCultureIgnoreCase));
}
```

Ketiga autocomplete tersebut menggunakan Data Source yang sama dan bersifat static. Untuk contoh diatas kita gunakan array string sebagai data source nya. Seperti pada sintaks dibawah.

```
private string value1, value2, value3;
private string[] states =
  {
   "ACEH", "SUMATERA UTARA", "SUMATERA BARAT", "RIAU",
   "JAMBI", "SUMATERA SELATAN", "BENGKULU", "LAMPUNG",
   "KEPULAUAN BANGKA BELITUNG", "KEPULAUAN RIAU", "DKI JAKARTA",
   "JAWA BARAT", "JAWA TENGAH", "DI YOGYAKARTA", "JAWA TIMUR", "BANTEN",
   "BALI", "NUSA TENGGARA BARAT", "NUSA TENGGARA TIMUR", "KALIMANTAN BARAT",
   "KALIMANTAN TENGAH", "KALIMANTAN SELATAN", "KALIMANTAN TIMUR",
   "KALIMANTAN UTARA", "SULAWESI UTARA",
   "SULAWESI TENGAH", "SULAWESI SELATAN", "SULAWESI TENGGARA", "GORONTALO",
   "SULAWESI BARAT", "MALUKU", "MALUKU UTARA", "PAPUA BARAT",
   "PAPUA"
};
```

2. Web Api



Pada bagian ini akan dijelaskan bagaimana jika menggunakan Web Api sebagai Data Sourcenya. Sebagai latihan kita gunakan web api yang sudah tersedia di internet seperti berikut (http://www.emsifa.com/api-wilayah-indonesia/api/provinces.json).

[{"id":"11","name":"ACEH"},{"id":"12","name":"SUMATERA UTARA"},{"id":"13","name":"SUMATERA BARAT"},{"id":"14","name":"RIAU"},
{"id":"15","name":"JAMBI"},{"id":"16","name":"SUMATERA SELATAN"},{"id":"17","name":"BENGKULU"},{"id":"18","name":"LAMPUNG"},
{"id":"19","name":"KEPULAUAN BANGKA BELITUNG"},{"id":"21","name":"KEPULAUAN RIAU"},{"id":"31","name":"DKI JAKARTA"},
{"id":"32","name":"JAWA BARAT"},{"id":"33","name":"JAWA TENGAH"},{"id":"34","name":"DI YOGYAKARTA"},{"id":"35","name":"JAWA
TIMUR"},{"id":"36","name":"BANTEN"},{"id":"51","name":"BALI"},{"id":"52","name":"NUSA TENGGARA BARAT"},{"id":"53","name":"NUSA
TENGGARA TIMUR"},{"id":"61","name":"KALIMANTAN BARAT"},{"id":"62","name":"KALIMANTAN TENGAH"},{"id":"63","name":"KALIMANTAN
SELATAN"},{"id":"64","name":"KALIMANTAN TIMUR"},{"id":"65","name":"KALIMANTAN UTARA"},{"id":"71","name":"SULAWESI UTARA"},
{"id":"72","name":"SULAWESI TENGAH"},{"id":"73","name":"SULAWESI SELATAN"},{"id":"74","name":"SULAWESI TENGGARA"},
{"id":"75","name":"GORONTALO"},{"id":"76","name":"SULAWESI BARAT"},{"id":"81","name":"MALUKU"},{"id":"82","name":"MALUKU
UTARA"},{"id":"91","name":"PAPUA BARAT"},{"id":"94","name":"PAPUA"}]

Gambar diatas adalah hasil dari api diatas. Terdapat dua buah property yaitu id dan name. Untuk itu perlu kita buah sebuah class dengan nama Provinsi dengan property seperti dibawah.

```
public class Provinsi
{
    public string id { get; set; }
    public string name { get; set; }
}
```

Untuk sintaks detail dari service dalam mengambil data pada web api dapat dilihat pada Lampiran Project. Artikel ini hanya menjelaskan Blazor dan MudBlazor saja. Tambahkan sintaks dibawah pada Blazor.

```
[Inject]
public IProvinsiRepository ProvinsiRepository { get; set; }
private Provinsi provinsiValue;
private async Task<IEnumerable<Provinsi>> SearchProvinsi(string value)
{
    var provinsi = (await ProvinsiRepository.GetAll()).ToList();
    if (string.IsNullOrEmpty(value)) return provinsi;

    return provinsi.Where(x => x.name.Contains(value,
    StringComparison.InvariantCultureIgnoreCase));
}
```

Selanjutnya tambahkan komponen seperti dibawah.

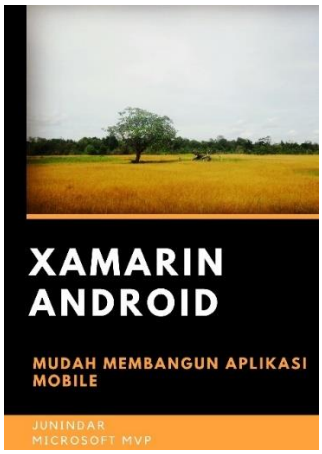
```
<MudAutocomplete T="Provinsi" Label="Provinsi" @bind-Value="provinsiValue"
ResetValueOnEmptyText="true" SearchFunc="@SearchProvinsi"
ToStringFunc="@(e=> e==null?null : $"{e.name}")" />
```

Untuk SearchFunc kita panggil function yang telah kita buat sebelumnya (SearchProvinsi). Lalu didalamnya kita panggil function lain untuk mendapatkan data dari web api. Selanjutnya baru melakukan pencarian berdasarkan nilai yang terdapat pada parameter “value”. Jika parameter “value” bernilai kosong atau null, maka tidak perlu melakukan pencarian lebih lanjut, langsung mengirikan data yang didapat dari web api.

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/2022/12/mudblazor-input-component-pada-blazor.html

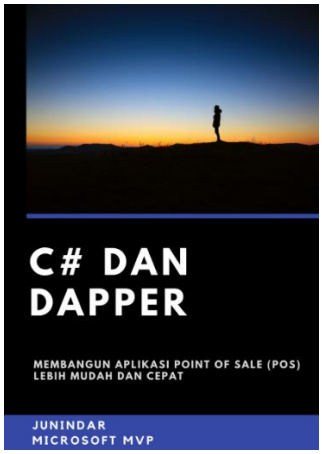
Referensi



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

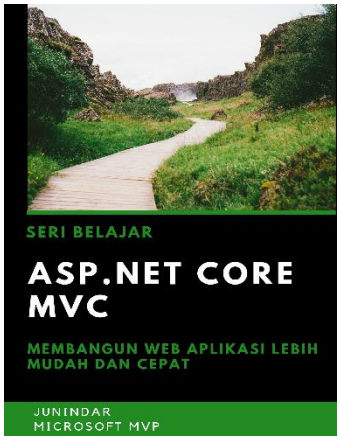
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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

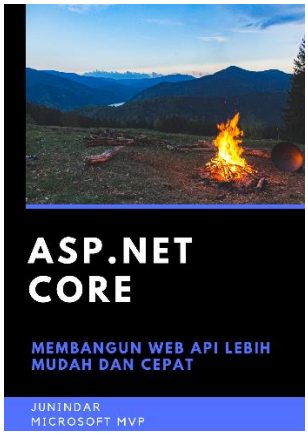


https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

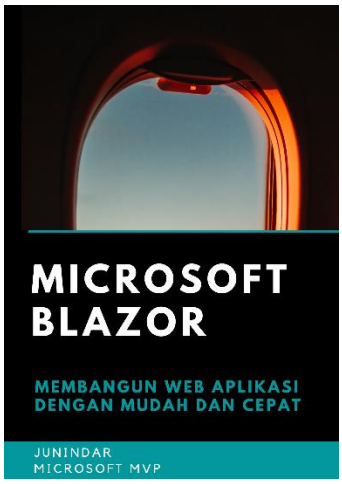
https://play.google.com/store/books/details/Junindar_ASP_NET_MVC_Membangun_Aplikasi_Web_Lebih?id=XLlyDwAAQBAJ



https://play.google.com/store/books/details/Junindar_ASP_NET_CORE_MVC?id=xEe5DwAAQBAJ



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".**