IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

JavaScript Interop Pada Aplikasi Blazor – Part 4

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 .

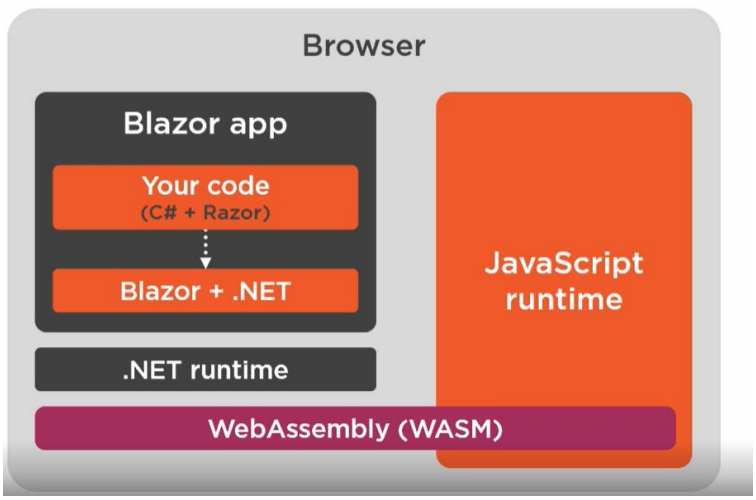
Code .Net berjalan pada WebAssembly, yang artinya kita dapat menjalankan “NET“ didalam browser (Client) tanpa harus menginstall plugin seperti Silverlight, Java mapun Flash.

Pendahuluan

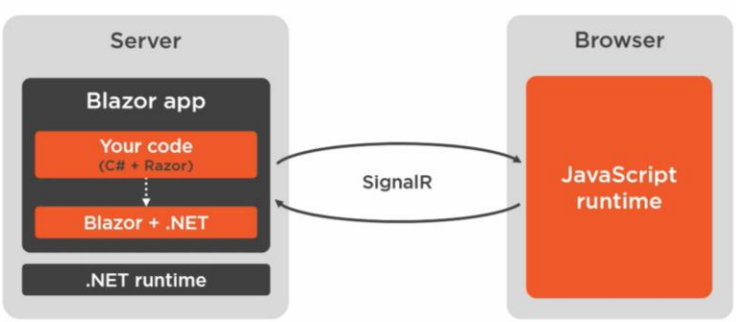
Untuk membuat aplikasi pada Blazor, kita menggunakan C# dan Razor. Razor merupakan kombinasi dari HTML dan C#. Dan output dari blazor aplikasi di eksekusi oleh .Net runtime.

Seperti kita ketahui, terdapat dua model hosting pada aplikasi blazor, yang pertama WebAssembly dan yang kedua adalah Server.

Untuk WebAssembly aplikasi dan .Net runtime berjalan pada sisi client didalam web browser. .Net runtime yang digunakan pada browser berdasarkan WebAssembly atau yang biasa disebut WASM. WASM adalah instruksi berformat binary yang dieksekusi Javascript runtime didalam browser. Jadi ini merupakan cara kerja dari Client Side hosting model pada blazor.

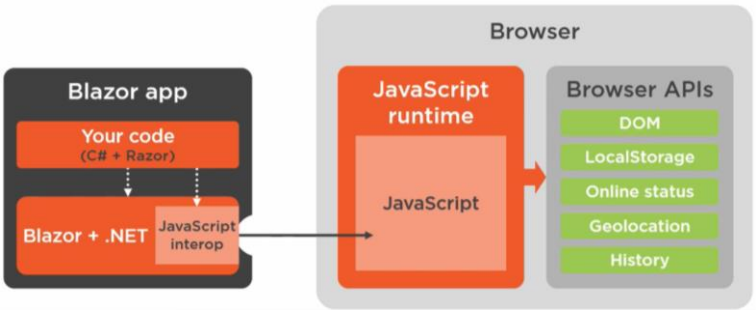


Kita dapat juga menjalankan blazor tanpa menggunakan WebAssembly, yaitu dengan menggunakan Server-Side hosting model. Yang artinya aplikasi blazor tidak dijalankan didalam browser, tetapi oleh server. Untuk melakukan render user interface pada browser, aplikasi berkomunikasi melalui SignalR dengan JavaScript runtime.



Browser juga memiliki browser API yang berbeda-beda, seperti Document Object Model (DOM). Dengan menggunkan DOM kita dapat mengkases dan mengganti elemen HTML pada aplikasi web. Browser API, seperti DOM ini dapat diakses dengan menggunakan JavaScript Runtime. Yang perlu diketahui, tanpa JavaScript Interop kita hanya dapat menggunakan fungsi yang hanya disediakan oleh Blazor Framework dan .Net. Lalu bagaimana jika kita ingin mengakses browser API dari code yang tidak disediakan oleh Blazor Framework? Untuk hal ini kita perlu memanggil code pada JavaScript yang akan mengakses Browser Api.

Blazor mendukung JavaScript Interoperability (JavaScript Interop), dimana kita dapat mengakses code pada JavaScript. Dari sini dapat kita ketahui, kapan kita harus menggunakan JavaScript pada aplikasi Blazor. Dimana jika aplikasi kita menggunakan fungsi-fungsi Browser API seperti DOM, Local Storage, Online Status yang tidak disediakan oleh Blazor Framework.



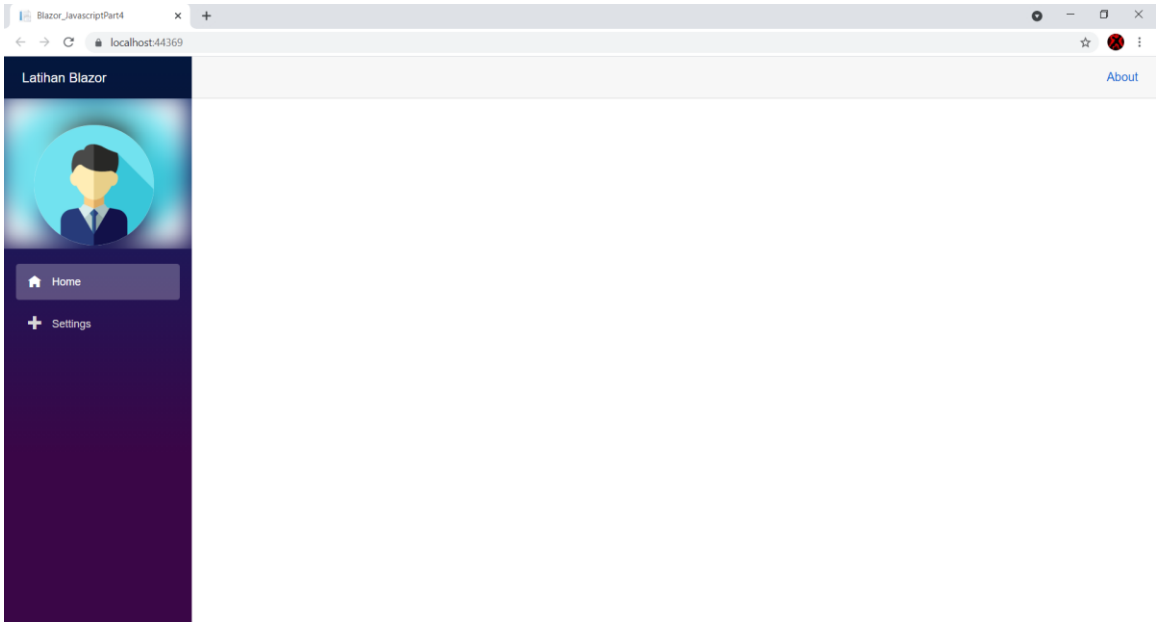
Pada artikel sebelumnya telah dibahas bagaimana melakukan invoke .Net method dari javascript. Untuk artikel ini akan dilanjutkan dengan integrasi browser API pada Blazor APP. Salah satu browser API yang akan menjadi studi kasus kita adalah web storage.

Apa itu web storage? Web Storage adalah tempat penyimpanan data pada local browser. Sebelum adanya HTML5, data pada aplikasi yang disimpan pada local browser menggunakan cookie. Dengan menggunkan Web Storage, kita dapat menyimpan data lebih besar dan lebih aman tanpa mempengaruhi kinerja dari aplikasi. Terdapat dua tipe dari web storage, yang pertama localStorage dan yang kedua adalah sessionStorage.

Untuk localStorage data yang disimpan tidak memiliki expiry date, dan data tidak akan dihapus walaupun browser telah ditutup. Berikut contoh penggunaan localStorage.

```
// Store
localStorage.setItem("lastname", "Junindar");
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
//delete
localStorage.removeItem("lastname");
```

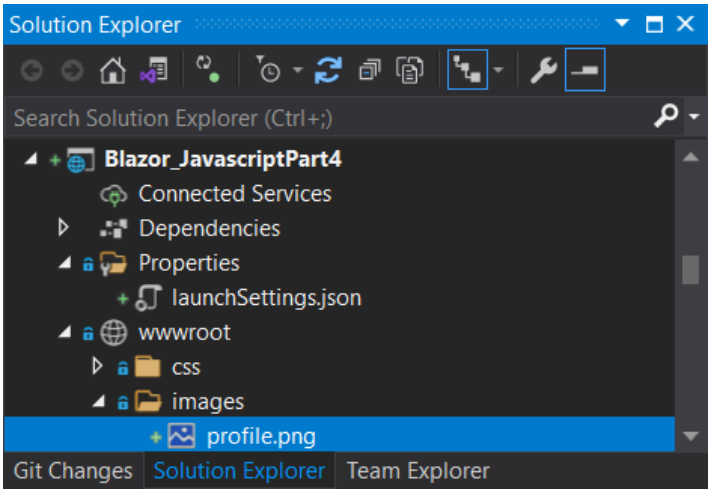
Sedangkan sessionStorage memiliki fungsi yang hampir sama dengan localStorage, kecuali data yang disimpan hanya untuk satu session (per session) dan data akan dihapus ketika pengguna menutup browser. Pada artikel ini kita hanya akan menggunakan localStorage sebagai studi kasus.



Pada latihan ini kita akan membuat halaman web sesuai dengan design diatas. Hasil akhir yang diinginkan dari artikel ini adalah, kita dapat menyimpan sekaligus mengganti Title dari aplikasi dan image profile.

Untuk memudahkan dalam memahami artikel ini, buat sebuah project Blazor App dan ikuti langkah-langkah dibawah ini.

- Tambahkan sebuah folder dengan nama "images" pada wwwroot, dan masukkan sebuah image untuk profile.



- Lalu ganti sintaks "NavMenu.razor" pada folder "Shared". Untuk detail sintaks dapat dilihat pada project lampiran.

```
<div class="top-row pl-4 navbar navbar-dark">
    <a class="navbar-brand" href="">@appTitle</a>
    <button class="navbar-toggler" @onclick="ToggleNavMenu">
        <span class="navbar-toggler-icon"></span>
    </button>
</div>
```

Pada code block tambahkan sebuah field "appTitle" seperti dibawah.

```
private string appTitle="Latihan Blazor";
```

Sedangkan untuk sintaks profile image, dapat dilihat seperti dibawah ini.

```
<div class="profile">
    <div class="profile-bg">
                <img class="img-responsive" src="images/profile.png" />
    </div>
    <div class="profile-picture">
        <img class="img-responsive circle account-photo" src="images/profile.png" />
    </div>
</div>
```

Lalu jalankan program, pastikan mendapatkan hasil seperti pada gambar diatas.

Note : Copy juga sintaks css pada file "css/site.css".

Setelah selesai dengan langkah-langkah diatas, kita lanjutkan pada backend, dengan membuat service-service yang diperlukan.

- Buat sebuah folder dengan nama "Entity", lalu tambahkan sebuah class seperti dibawah ini.

```
public class Setting
{
   public string AppTitle { get; set; }

   public string UserPictureUrl { get; set; }
}
```

- Tambahkan sebuah folder dengan nama "Service", dan buat 2 buah folder dengan nama masing-masing "LocalStorage" dan "Settings". Pada folder Settings tambahkan sebuah class dengan nama "SettingsChangedEventArgs" dan ketikkan detail sintaksnya seperti dibawah.

```
public class SettingsChangedEventArgs : EventArgs
{
      public SettingsChangedEventArgs(Setting setting)
      {
            Setting = setting;
      }

      public Setting Setting { get; }
}
```

  Class ini menggunakan class "EventArgs" yang merupakan base class untuk semua event pada class.

- Selanjutnya tambahkan sebuah interface dengan nama "ISettingsService", dan ketikkan detail sintaksnya seperti dibawah. Pada interface ini terdapat sebuah property dari class Setting, event dan method.

```
public  interface ISettingsService
{
     Setting Setting { get; set; }
     event EventHandler<SettingsChangedEventArgs> SettingsChanged;
     void RaiseSettingsChanged();
}
```

- Kita lanjutkan dengan membuat class "SettingsService" yang mengimplementasikan interface diatas.

  Untuk property Setting, kita buat terlebih defaul valuenya. Sehingga jika data pada localStorage masih kosong, maka nilai ini yang akan digunakan.

```
public class SettingsService: ISettingsService
{
    public Setting Setting { get; set; } = new Setting
    {
         AppTitle = "Latihan Blazor",
         UserPictureUrl = "images/profile.png",

    };

    public event EventHandler<SettingsChangedEventArgs> SettingsChanged;
    public void RaiseSettingsChanged()
    {
        SettingsChanged?.Invoke(this, new SettingsChangedEventArgs(Setting));
   }
}
```

Untuk method "RaisedSettingsChanged" menggunana event "SettingChanged" yang memiliki paramater class Setting.

Setelah selesai dengan service diatas, kita lanjutkan dengan membuat halaman Settings dengan menambahkan sebuah razor component dengan nama "Settings.razor". Untuk detail sintaks html-nya dapat dilihat pada project lampiran.

Note : Pastikan service diatas di register pada startup.cs (`ConfigureServices`).

```
services.AddSingleton<ISettingsService, SettingsService>();
```

- Tambahkan sintaks berikut pada code block

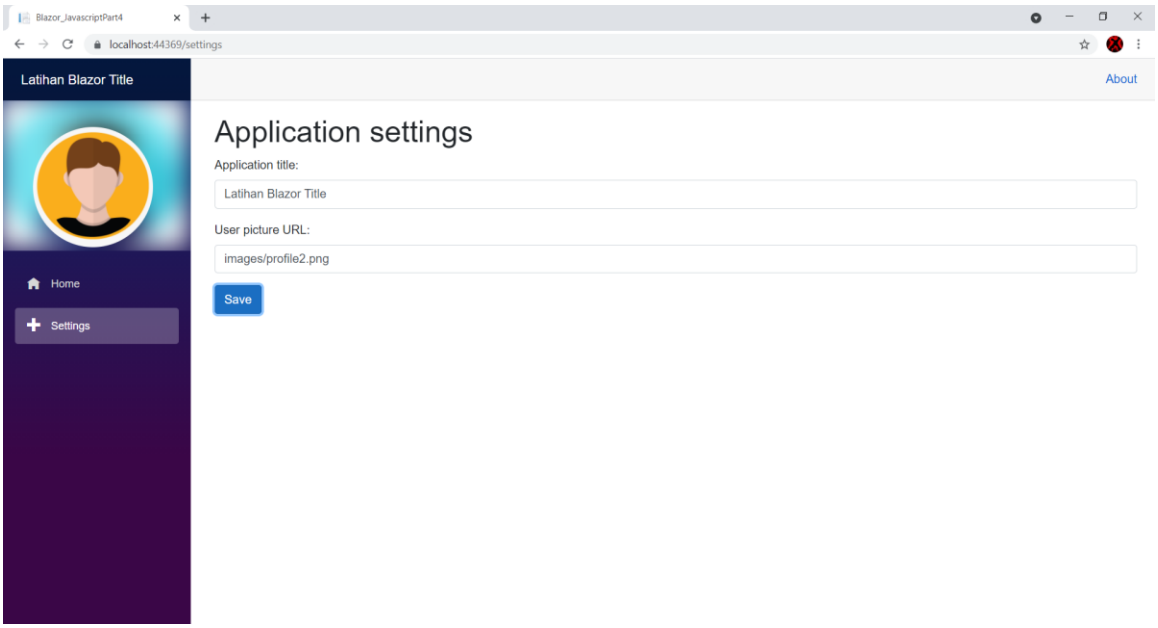
```
Setting settings;

protected override void OnInitialized()
{
   settings = SettingsService.Setting;
}

async Task SaveSettingsAsync()
{
    SettingsService.RaiseSettingsChanged();
}
```

Pada method OnInitilized kita panggil SettingService.Setting untuk mengambil data Setting yang nantinya akan ditampilkan pada page settings. Sedangkan method "SaveSettingAsync" digunakan pada button Save (OnClick).

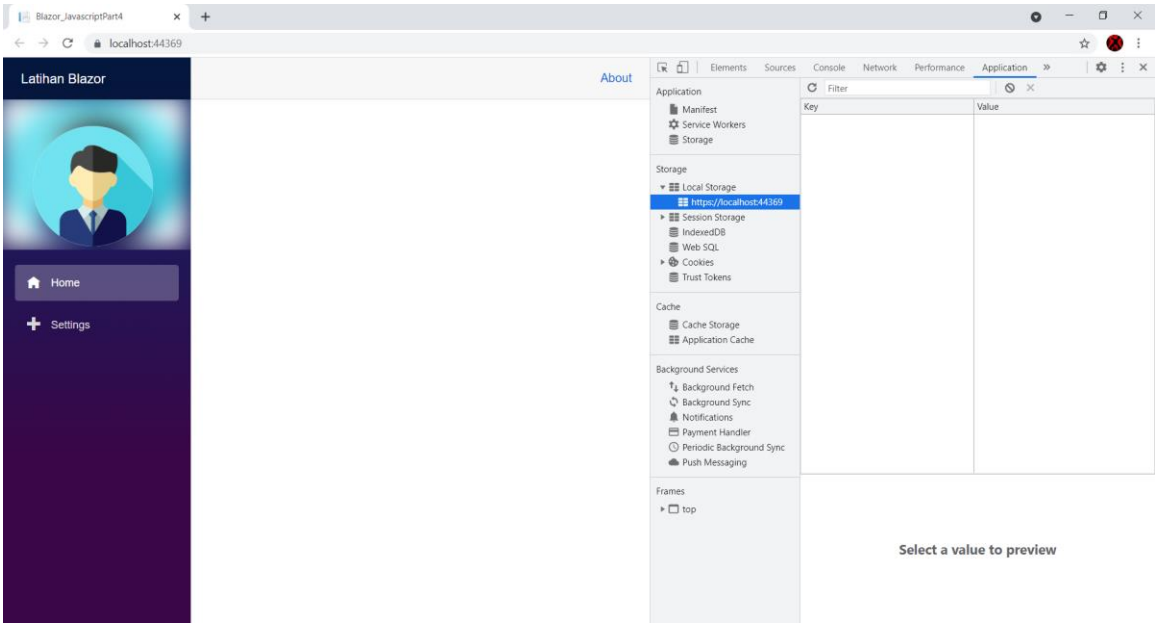
Selanjutnya buka file "NavMenu.razor" dan tambahkan sintak dibawah ini.

```
private string appTitle="Latihan Blazor";
private Setting settings;
protected override void OnInitialized()
{
        SettingsService.SettingsChanged += SettingsChanged;
        settings = SettingsService.Setting;
        appTitle = settings.AppTitle;
}
public void Dispose()
{
        SettingsService.SettingsChanged -= SettingsChanged;
}
private void SettingsChanged(object sender, SettingsChangedEventArgs e)
{
  this.settings = e.Setting;
  appTitle = settings.AppTitle;
  InvokeAsync(() =>
  {
      StateHasChanged();
  });
}
```

Jalankan program dan buka page settings. Coba ganti value untuk title maupun image url dan tekan button Save, pastikan Application Title dan Image Profile pada menu berubah juga.



Coba tutup aplikasi dan jalankan kembali lalu liat apa yang terjadi. Title dan Image kembali seperti semula.



Buka browser Developer Tools (F12) dan klik Application Tab. Dapat kita lihat dimana pada local storage tidak ada data sama sekali. Disini akan kita lanjutkan untuk membuat service untuk menyimpan dan mengambil data pada local storage.

- Buat sebuah interface seperti dibawah ini pada folder LocalStorage.

```
public interface ILocalStorageService
{
    Task SetItemAsync<T>(string key, T item);

    Task<T> GetItemAsync<T>(string key);
}
```

- Lalu dilanjutkan dengan membuat service untuk mengimplementasikan interface diatas.

  Pada class ini terdapat dua buah method yang memanggil javascript function. ("`localStorage.setItem`" dan "`localStorage.getItem`"). Pada method GetItemAsync terdapat sebuah kondisi, dimana jika tidak ada data pada localStorage yang dicari maka nilai yang akan digunakan adalah "default"

```
public class LocalStorageService : ILocalStorageService
{
      private readonly IJSRuntime _jsRuntime;

      public LocalStorageService(IJSRuntime jsRuntime)
      {
          _jsRuntime = jsRuntime;
      }

      public async Task SetItemAsync<T>(string key, T item)
      {
        await _jsRuntime.InvokeVoidAsync("localStorage.setItem",
                      key, JsonSerializer.Serialize(item));
      }

      public async Task<T> GetItemAsync<T>(string key)
      {
        var json = await _jsRuntime.InvokeAsync<string>("localStorage.getItem", key);
                  return string.IsNullOrEmpty(json)
                      ? default
                      : JsonSerializer.Deserialize<T>(json);
      }
}
```

Lalu register service diatas pada startup.cs (`ConfigureServices`) seperti dibawah.

```
services.AddTransient<ILocalStorageService, LocalStorageService>();
```

Selanjutnya buka kembali file Settings.razor, dan tambahkan sintaks untuk memanngil method "SetItemAsync" dari class LocalStoreage pada method "`SaveSettingsAsync`", seperti dibawah ini.

```
await LocalStorageService.SetItemAsync("settings", settings);
```

Dan terakhir kita tambahkan sintaks seperti dibawah pada App.Razor

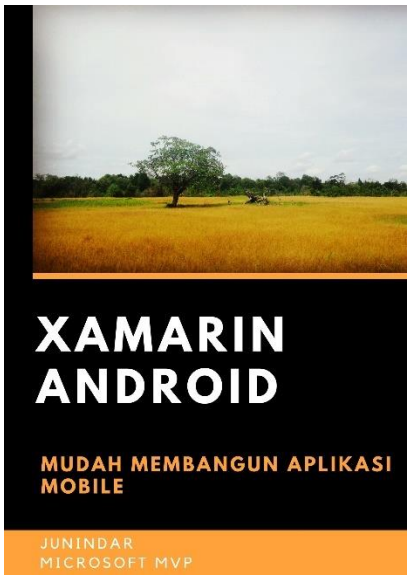
```
protected override async Task OnAfterRenderAsync(bool firstRender)
{
   if (firstRender)
   {
       await InitializeUserSettingsFromLocalStorageAsync();
   }
}

private async Task InitializeUserSettingsFromLocalStorageAsync()
{
   var userSettings = await LocalStorageService.GetItemAsync<Setting>("settings");
   if (userSettings != null)
   {
       SettingsService.Setting = userSettings;
       SettingsService.RaiseSettingsChanged();
   }
}
```

Jalankan program dan ganti value-value pada page Settings dan Save. Tutup browser dan jalankan kembali. Pastikan Application Title dan Profile Image tidak kembali menjadi seperti semula. Dan buka Developer Tools untuk mengecek localStorage.

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/2021/11/javascript-interop-pada-aplikasi-blazor.html

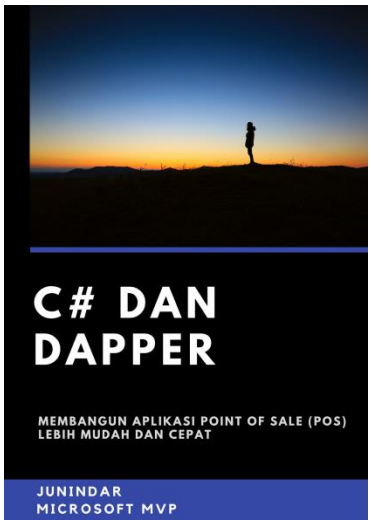
Referensi



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

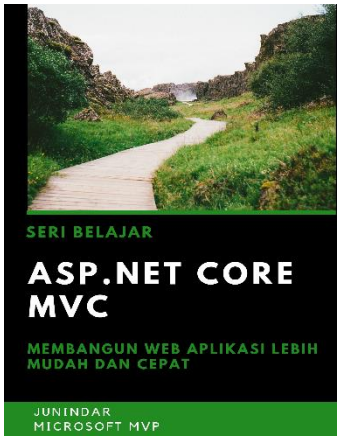
https://play.google.com/store/books/details?id=VSLiDQAAQBAJ

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

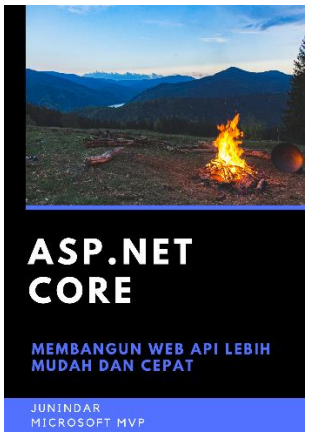


https://play.google.com/store/books/details/Junindar_C_dan_Dapper_Membangun_Aplikasi_POS_Point?id=6TErDwAAQBAJ

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

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".**