

C#으로 웹UI 프로그래밍

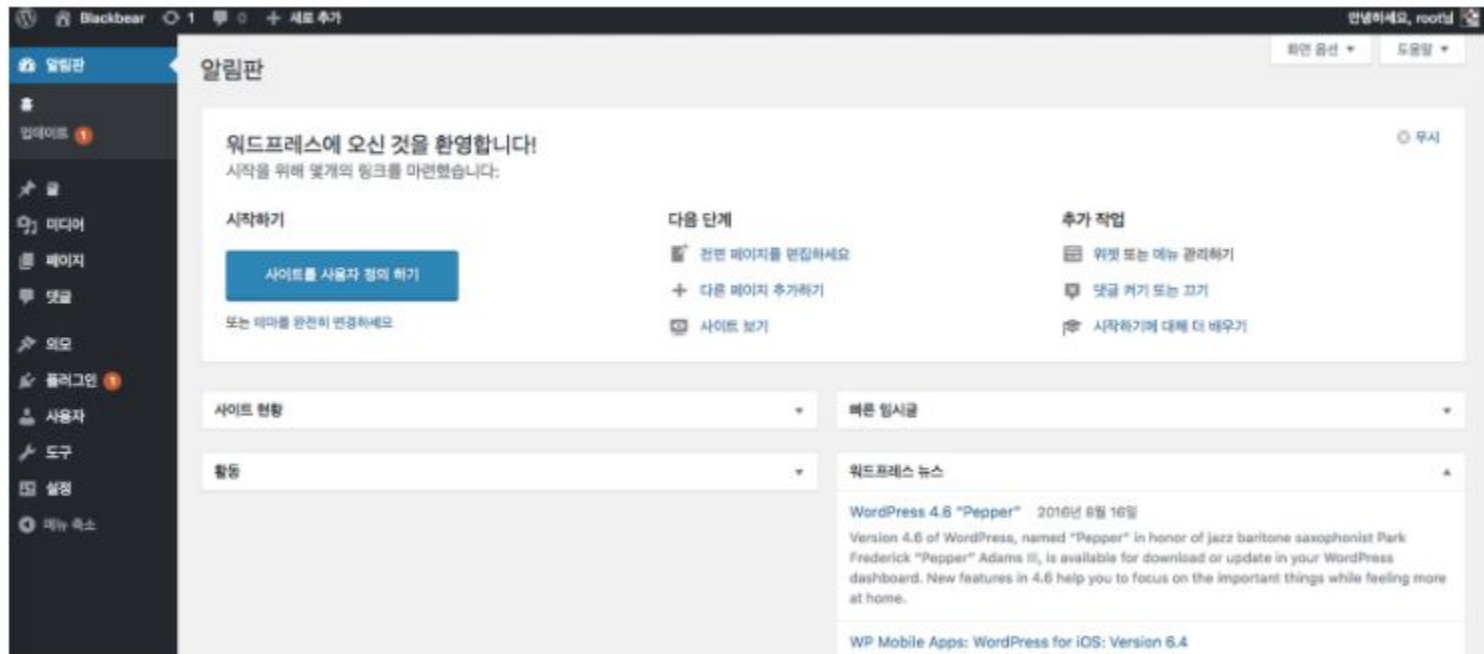
컴투스 센트럴서버실
최흥배

게임 서버 개발을 할 때는 **게임 서버**를 만드는 것이 핵심이지만 효율적인 개발과 운영을 위해서는 **툴**도 꼭 만들어야 한다.

- 게임 리소스 입력
- DB 데이터 변경
- 게임 서버 모니터링
- 로그 분석
- 배포

등등

10년 전에는 이런 툴을 PC용 데스크탑 프로그램으로 만들기도 했지만
이제는 특수한 경우가 아니라면 당연히 웹(Web)이다.



게임 서버 개발자에게
Web API 서버 개발은 어렵지 않지만
HTML + JavaScript로 이루어진 웹 프론트엔드
개발은 매우 어려움;;;

게임 서버 개발자가 하기 난감한 웹 프론트엔드 개발을
HTML, JavaScript가 아닌 C#으로 개발할 수가 있음.

C#으로 GUI 작업을 할 때는 WinForm 혹은 WPF로
하는데 이 2가지 기술 경험으로 웹 프론트엔드 개발을
할 수가 있다.

Blazor
OpenSilver

웹어셈블리의 컨셉

본 글에서는 웹어셈블리의 작동원리 뒤에 숨어있는 컨셉을 설명함과 동시에 웹어셈블리의 목표, 웹어셈블리가 해결할 수 있는 문제, 그리고 웹브라우저 렌더링 엔진 안에서 웹어셈블리가 작동하는 원리에 대해 설명하려고 합니다.

웹어셈블리가 뭔가요?

WebAssembly는 최신 웹 브라우저에서 실행할 수 있는 새로운 유형의 코드이며 새로운 기능과 성능 면에서 큰 이점을 제공합니다. 직접 코드를 작성하는 것이 아니라 C, C++, RUST 등의 저급 소스 언어를 효과적으로 컴파일하도록 고안되었습니다.

이는 웹 플랫폼에 큰 영향을 미칩니다. 이전에 불가능했던 웹에서 실행되는 클라이언트 응용 프로그램을 사용하여 웹에서 여러 언어로 작성된 코드를 네이티브에 가까운 속도로 실행하는 길을 제공합니다.

게다가 WebAssembly 코드를 사용하여 이를 활용하는 방법을 알 필요조차 없습니다. WebAssembly 모듈을 웹 (또는 Node.js) 앱으로 가져와 JavaScript를 통해 사용할 수 있도록 할 수 있습니다. JavaScript 프레임 워크는 WebAssembly를 사용하여 대규모 성능 이점과 새로운 기능을 제공하면서도 웹 개발자가 쉽게 기능을 사용할 수 있도록 할 수 있습니다.

<https://developer.mozilla.org/ko/docs/WebAssembly/Concepts>

ASP.NET Core Blazor 소개

2021. 10. 08. • 읽는 데 17분 걸림 • 🦎 🍷

Blazor에 오신 것을 환영합니다!

Blazor는 .NET을 사용하여 대화형 클라이언트 쪽 웹 UI를 빌드하기 위한 프레임워크입니다.

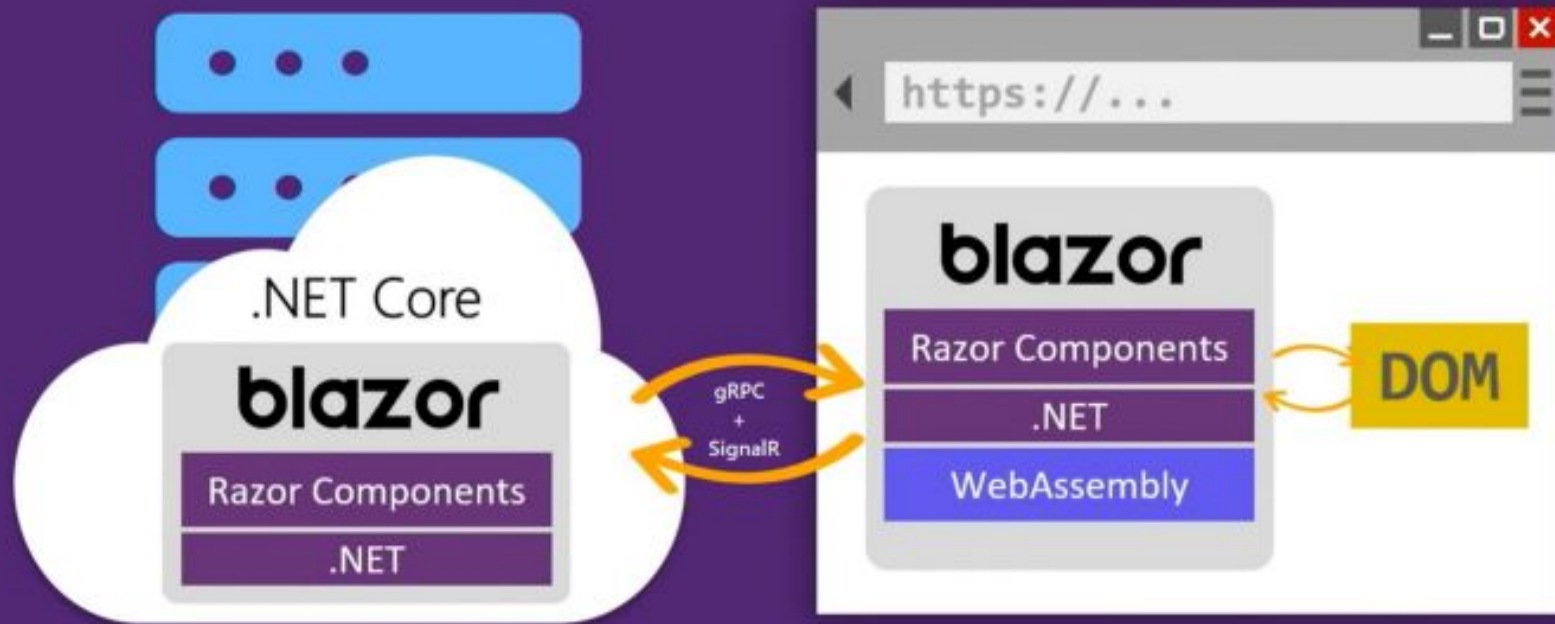
- JavaScript 대신 C#을 사용하여 풍부한 대화형 UI를 만듭니다.
- .NET에서 작성된 서버 쪽 및 클라이언트 쪽 앱 논리를 공유합니다.
- 모바일 브라우저를 포함한 광범위한 브라우저 지원을 위해 UI를 HTML 및 CSS로 렌더링합니다.
- Docker와 같은 최신 호스트 플랫폼과 통합합니다.

클라이언트 쪽 웹 개발에 .NET을 사용하면 다음과 같은 이점이 있습니다.

- JavaScript 대신 C#으로 코드를 작성합니다.
- .NET 라이브러리의 기존 .NET 에코시스템을 활용합니다.
- 서버 및 클라이언트에서 앱 논리를 공유합니다.
- .NET의 성능, 안정성 및 보안을 활용합니다.
- Windows, Linux 및 macOS에서 Visual Studio를 사용하여 생산성을 유지합니다.
- 안정적이고, 기능이 풍부하고, 사용하기 쉬운 공통 언어, 프레임워크 및 도구 세트를 기반으로 빌드합니다.

<https://docs.microsoft.com/ko-kr/aspnet/core/blazor/?view=aspnetcore-5.0>

Hybrid Blazor: switching Server and WebAssembly at runtime



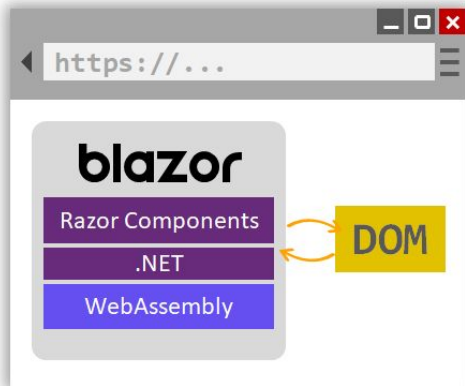
웹 호스팅 모델

Blazor WebAssembly 앱

Blazor WebAssembly 앱은 WebAssembly 기반 .NET 런타임의 브라우저에서 직접 실행됩니다. Blazor WebAssembly 앱은 Angular 또는 React 같은 프론트 엔드 JavaScript 프레임워크와 비슷한 방식으로 작동합니다. 그러나 JavaScript를 작성하는 대신 C#을 작성합니다. 앱 어셈블리 및 모든 필수 종속성과 함께 .NET 런타임이 앱과 함께 다운로드됩니다. 브라우저 플러그인 또는 추가 확장이 필요하지 않습니다.

다운로드된 어셈블리는 다른 .NET 앱에서 사용하는 것과 같은 일반적인 .NET 어셈블리입니다. 런타임은 .NET Standard를 지원하기 때문에 Blazor WebAssembly 앱에서 기존 .NET Standard 라이브러리를 사용할 수 있습니다. 그러나 해당 어셈블리는 여전히 브라우저 보안 샌드박스에서 실행됩니다. 일부 기능은 파일 시스템에 액세스하거나 임의 네트워크 연결을 여는 것처럼 `PlatformNotSupportedException`을 throw할 수 있습니다.

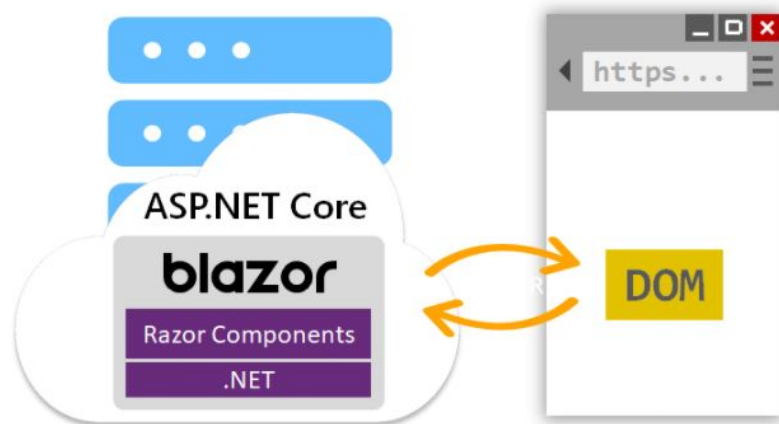
앱이 로드되면 .NET 런타임이 시작되고 앱 어셈블리를 가리킵니다. 앱 시작 논리가 실행되고 루트 구성 요소가 렌더링됩니다. Blazor는 구성 요소의 렌더링된 출력을 기반으로 UI 업데이트를 계산합니다. 그런 다음, DOM 업데이트가 적용됩니다.



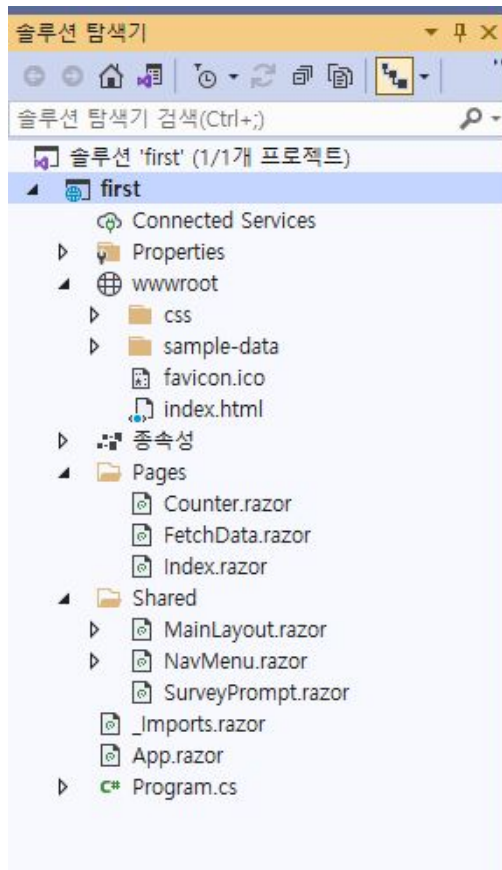
Blazor Server 앱

Blazor 아키텍처에서 설명한 대로 Blazor 구성 요소는 출력을 `RenderTree`라는 중간 추상화로 렌더링합니다. 그런 다음, Blazor 프레임워크는 렌더링된 항목을 이전에 렌더링된 항목과 비교합니다. 차이점이 DOM에 적용됩니다. Blazor 구성 요소는 렌더링된 출력을 적용하는 방식에서 분리됩니다. 따라서 구성 요소 자체는 UI를 업데이트하는 프로세스와 동일한 프로세스에서 실행할 필요가 없습니다. 실제로 동일한 머신에서 실행하지 않아도 됩니다.

Blazor Server 앱에서 구성 요소는 브라우저의 클라이언트 쪽이 아니라 서버에서 실행됩니다. 브라우저에서 발생하는 UI 이벤트는 실시간 연결을 통해 서버에 전송됩니다. 이벤트는 올바른 구성 요소 인스턴스로 디스패치됩니다. 구성 요소가 렌더링되고 계산된 UI 차이는 직렬화되어 DOM에 적용되는 브라우저로 전송됩니다.



Blazor 프로젝트 템플릿으로 시작하기



Blazor만 있으면 이제 웹 프론트엔드 개발이 쉬워졌을까?

아니다..... 여전히 **Blazor**만으로는 쉽지않다.

Blazor은 **JavaScript**를 **C#**으로 대체하는 것이라서 여전히 **HTML**, **CSS**로 웹UI 개발하는 부분은 헛갈리고 어렵다.

다행히 해결 방법은 있다!!!

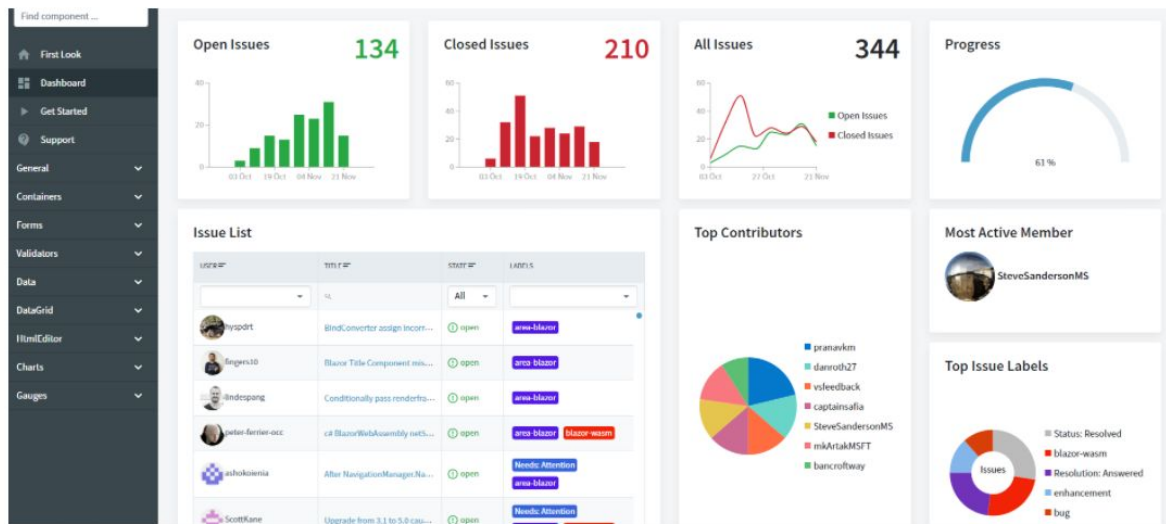
유니크하고 멋진 사이트를 만드는 것이 아니라면 **Blazor**용 오픈 소스 컴포넌트 라이브러리를 사용하면 웹UI 개발이 많이 쉬워진다.

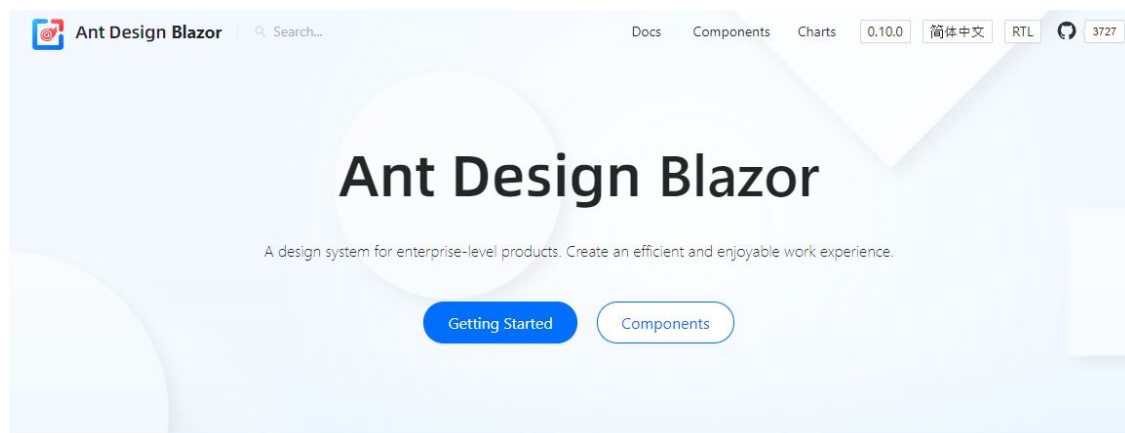
Top 10 nice free Blazor components

1. Radzen

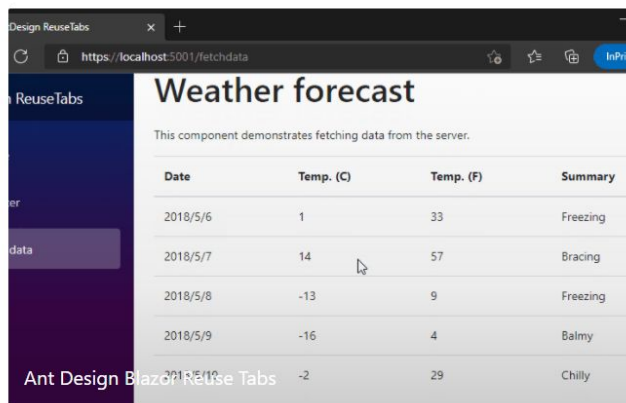
Radzen is a group of more than 60 Blazor components that facilitates the development of Dashboards, Intranet, and applications with similar purposes.

Link: <https://blazor.radzen.com/>



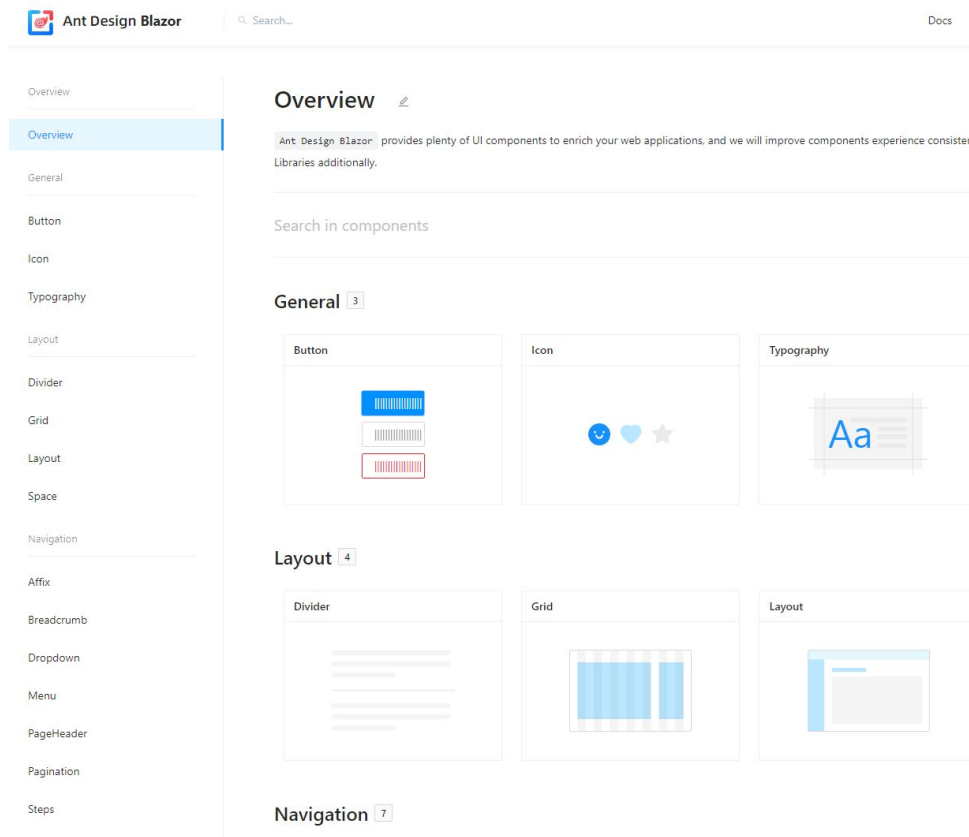


Recommended



<https://antblazor.com/en-US/>

어떤 컴포넌트가 있고, 어떻게 동작하는지 알고 싶다면 아래를 참고한다



<https://antblazor.com/en-US/components/overview>

Import Ant Design Blazor into an existing project

- Go to the project folder of the application and install the Nuget package reference

```
$ dotnet add package AntDesign
```

- Register the services in `Program.cs` (WebAssembly)

```
builder.Services.AddAntDesign();
```

or `Startup.cs` (Server)

```
services.AddAntDesign();
```

- Link the static files in `wwwroot/index.html` (WebAssembly) or `Pages/_Host.cshtml` (Server)

```
<link href="_content/AntDesign/css/ant-design-blazor.css" rel="stylesheet" />  
<script src="_content/AntDesign/js/ant-design-blazor.js"></script>
```

- Add namespace in `_Imports.razor`

```
@using AntDesign
```

<https://github.com/ant-design-blazor/ant-design-blazor#installation-guide>

샘플 예제 설명

Blazor와 함께 ASP.NET Core SignalR 사용

2021. 10. 14. • 읽는 데 41분 걸림 • 🦊 🍷

Choose a Blazor hosting model

Blazor Server

Blazor WebAssembly

이 자습서에서는 Blazor와 함께 SignalR을 이용해서 실시간 앱을 구현하기 위한 기본 사항을 알려 줍니다. 다음과 같은 작업을 수행하는 방법을 살펴봅니다.

- ✓ Blazor 프로젝트 만들기
- ✓ SignalR 클라이언트 라이브러리 추가
- ✓ SignalR 허브 추가
- ✓ SignalR 서비스 및 SignalR 허브에 대한 엔드포인트 추가
- ✓ 채팅을 위한 Razor 구성 요소 코드 추가

이 모든 과정을 마치면 채팅 앱을 실행할 수 있습니다.

<https://docs.microsoft.com/ko-kr/aspnet/core/tutorials/signalr-blazor?view=aspnetcore-5.0&tabs=visual-studio&pivots=server>



OpenSilver is a modern, plugin-free, open-source reimplementation of Silverlight. It uses Mono for WebAssembly and Microsoft Blazor to bring back the power of C#, XAML, and .NET to client-side Web development.

Reuse your codebase. Improve your app. Run in every browser.

[DOWNLOAD NOW](#)

v1.0.0 - Requires Visual Studio 2019 (16.11+) or 2022 for Windows

<https://opensilver.net/>

develop 68 branches 66 tags

Github1s

Go to file

Add file

Code

barjonp fix: convert DropShadowEffect to JavaScript with invariant culture 47b7f42 10 hours ago 1,370 commits

build	chore: update .bat files to generate OpenSilver's Runtime and Simulat...	2 days ago
libs/BridgeNET	Rename folder "ExtLibs" to "libs"	14 months ago
src	fix: convert DropShadowEffect to JavaScript with invariant culture	10 hours ago
.gitignore	VSIX: Cleaned up OpenSilver templates.	5.77 KB 4 months ago
CONTRIBUTING.md	Rename Contributing.md.	3.64 KB 6 months ago
LICENSE.txt	Updated copyrights	1.04 KB 4 months ago
README.md	chore: merge all WORKINPROGRESS members into default configuratio...	4.2 KB last month
THIRD-PARTY-NOTICES.t...	chore: add the .NET Runtime license to third-party notice	26.69 KB last month
restore-packages-cshtm...	chore: merge all WORKINPROGRESS members into default configuratio...	281 Bytes last month
restore-packages-opens...	chore: bump package used to build runtime to version 1.0.0	267 Bytes 2 days ago

README.md

This repository contains the source code of the 2 following products, which share most of their code:

- **OpenSilver** (www.opensilver.net) → It compiles C#/XAML/.NET to WebAssembly/HTML/CSS (Free, Open Source, MIT Licensed)
- **CSHTML5** (www.cshtml5.com) → It compiles C#/XAML/.NET to JavaScript/HTML/CSS (Dual Licensed)

The main branches are:

- **develop**: this branch is where day to day development occurs
- **master**: this branch corresponds to the version of the packages that are on Nuget.org

<https://github.com/OpenSilver/OpenSilver>



Silverlight 지원 종료

Windows 7, Silverlight

Microsoft [Silverlight](#)는 2021년 10월 12일 지원이 종료됩니다. Silverlight 개발 프레임워크는 현재 Internet Explorer 10 및 Internet Explorer 11에서만 지원되며, Internet Explorer 10에 대한 지원은 2020년 1월 31일에 종료됩니다. Chrome, Firefox 또는 Mac 운영 체제를 사용하는 브라우저는 더 이상 지원되지 않습니다.

- 오래전 MS가 어도비의 플래시의 대항마로 만든 웹플러그인(ActiveX) 웹UI 툴 (혹은 프레임워크)로 **실버라이트** 라는 것을 만들었음.
 - 이미 몇년 전에 MS는 실버라이트를 포기했음
 - 또 플래시도 이제 역사에서 사라졌음
- **OpenSilver**는 아주 간단하게 설명하면 실버라이트의 오픈소스 버전이다.
 - OpenSilver is a modern, plugin-free, open-source reimplement of Silverlight
 - It uses Mono for WebAssembly and Microsoft Blazor to bring back the power of C#, XAML, and .NET to client-side Web development
- UI를 **XAML**로 프로그래밍 할 수 있다.
 - 기존에 실버라이트로 만든 것이나 혹은 WPF로 만든 것을 빠르게 포팅할 수 있다.
 - WPF 프로그래밍에 익숙하면 아주 쉽게 OpenSilver를 사용할 수 있다.
- OpenSilver는 C# + XAML로 만든 코드를 WebAssembly를 사용하여 웹UI 개발을 할 수 있게 해준다.

OpenSilver Gallery

OpenSilver Showcase

A sample application made with OpenSilver. It contains examples and code snippets that you can reuse to build your own apps.

This application is written entirely in C# and XAML, and compiled to WebAssembly using OpenSilver.

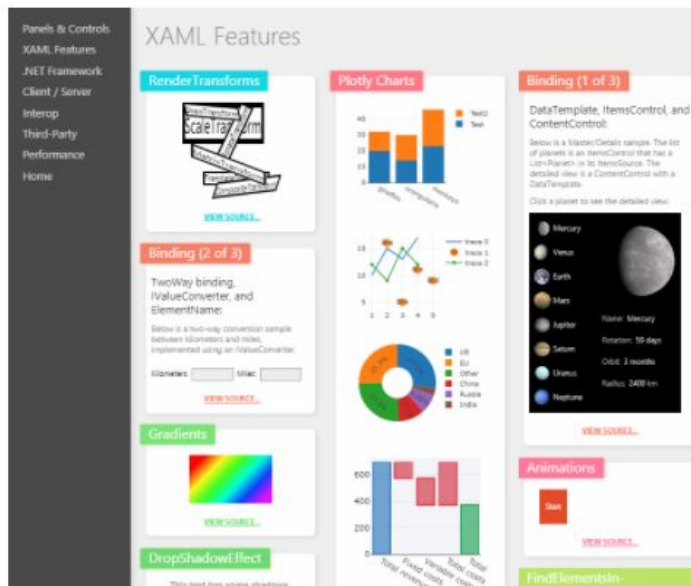
OpenSilver Showcase :

[View Live](#)

[See Source on GitHub](#)

Telerik UI Showcase :

[View Live](#)



<https://opensilver.net/gallery/>

Panels & Controls

Grid



[VIEW SOURCE...](#)

StackPanel



[VIEW SOURCE...](#)

DockPanel



[VIEW SOURCE...](#)

WrapPanel



[VIEW SOURCE...](#)

Canvas



[VIEW SOURCE...](#)

TextBox

Type your name:

John Doe

OK

[VIEW SOURCE...](#)

TextBlock

Use a TextBlock to display some text.

[VIEW SOURCE...](#)

CheckBox

☐ Check me!

[VIEW SOURCE...](#)

RadioButton

☐ Option 1 ☐ Option 2

[VIEW SOURCE...](#)

Button

This is a button -Click me!

[VIEW SOURCE...](#)

ToolTip

Button with text ToolTip inline

Button with image ToolTip as direct child

[VIEW SOURCE...](#)

ScrollView



[VIEW SOURCE...](#)

Border

Unlike a Rectangle, a Border can contain a child element (such as this text).

[VIEW SOURCE...](#)

Path (vector)



[VIEW SOURCE...](#)

Label

Non-Styled label

Styled label

[VIEW SOURCE...](#)

ComboBox



[VIEW SOURCE...](#)

GridSplitter



NumericUpDown

0 - +

[VIEW SOURCE...](#)

Slider



RepeatButton

Click and hold on the following buttons:

Translate Rotate



HyperlinkButton

You can visit the homepage by clicking here

[VIEW SOURCE...](#)

ListBox

Mercury
Venus
Earth

사용하기

- Visual Studio 2019(16.11+) 이상이 필요하다.
- Visual Studio용 애드인을 [다운로드](<https://opensilver.net/download.aspx>) 하고 설치한다.
- 새 프로젝트에서 OpenSilver용 템플릿을 선택한다.
- [Getting Started](#)

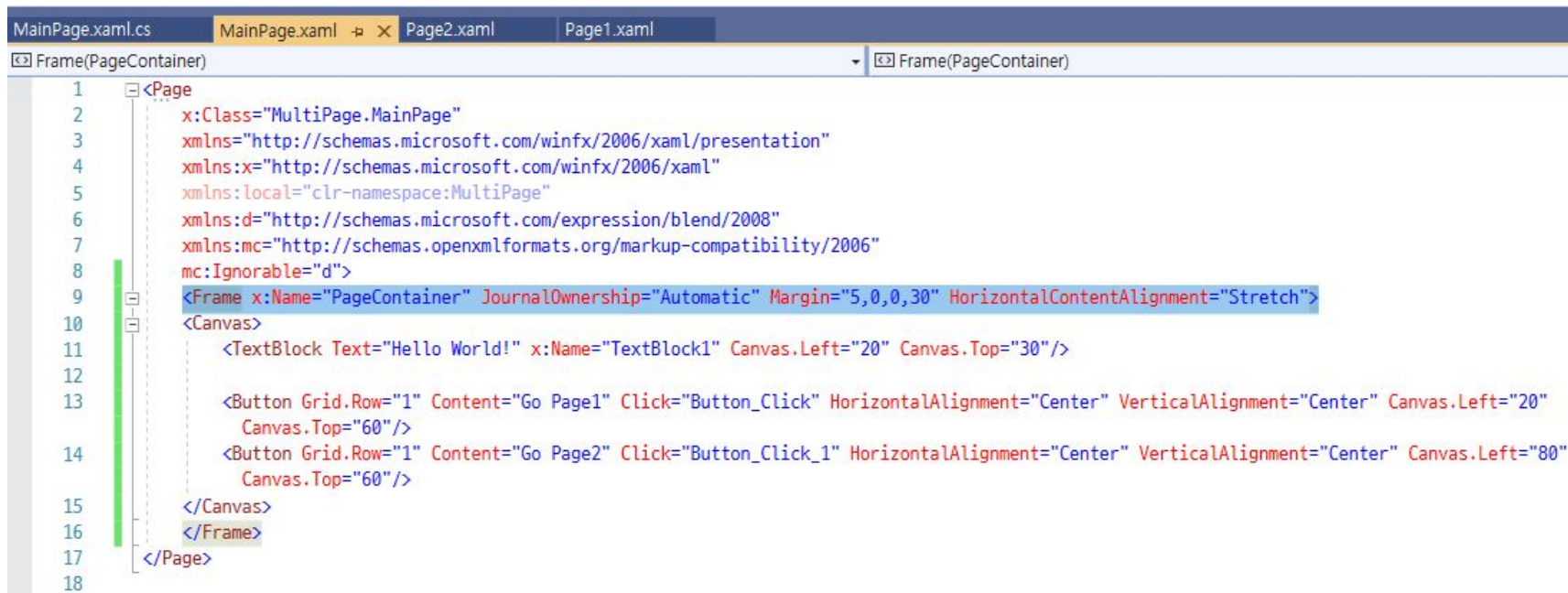
Blazor와 비교

- 좋은 점
 - WPF를 안다면 UI 배치가 훨씬 더 쉽다
 - WPF를 안다면 UI 동작에 대해서 따로 배울 필요가 거의 없다.
 - 시뮬레이터가 있어서 실제 어떻게 보일지 알 수 있다.
- 아쉬운 점
 - MS 정품이 아님
 - Visual Studio가 필수
 - 개발 시 서버 연동은 별도로 직접 만들어야 한다

샘플 예제 설명

Page 이동(화면 변경)

- Page 이동은 일반적인 웹 프론트 프로그래밍에서 .html이 바뀌는 것과 같다.
- OpenSilver은 XAML 기반이므로 하나의 화면의 Page로 정의한다
- 복수의 Page를 사용할 때는 메인 Page에는 **Frame** 키워드를 정의해야 한다.



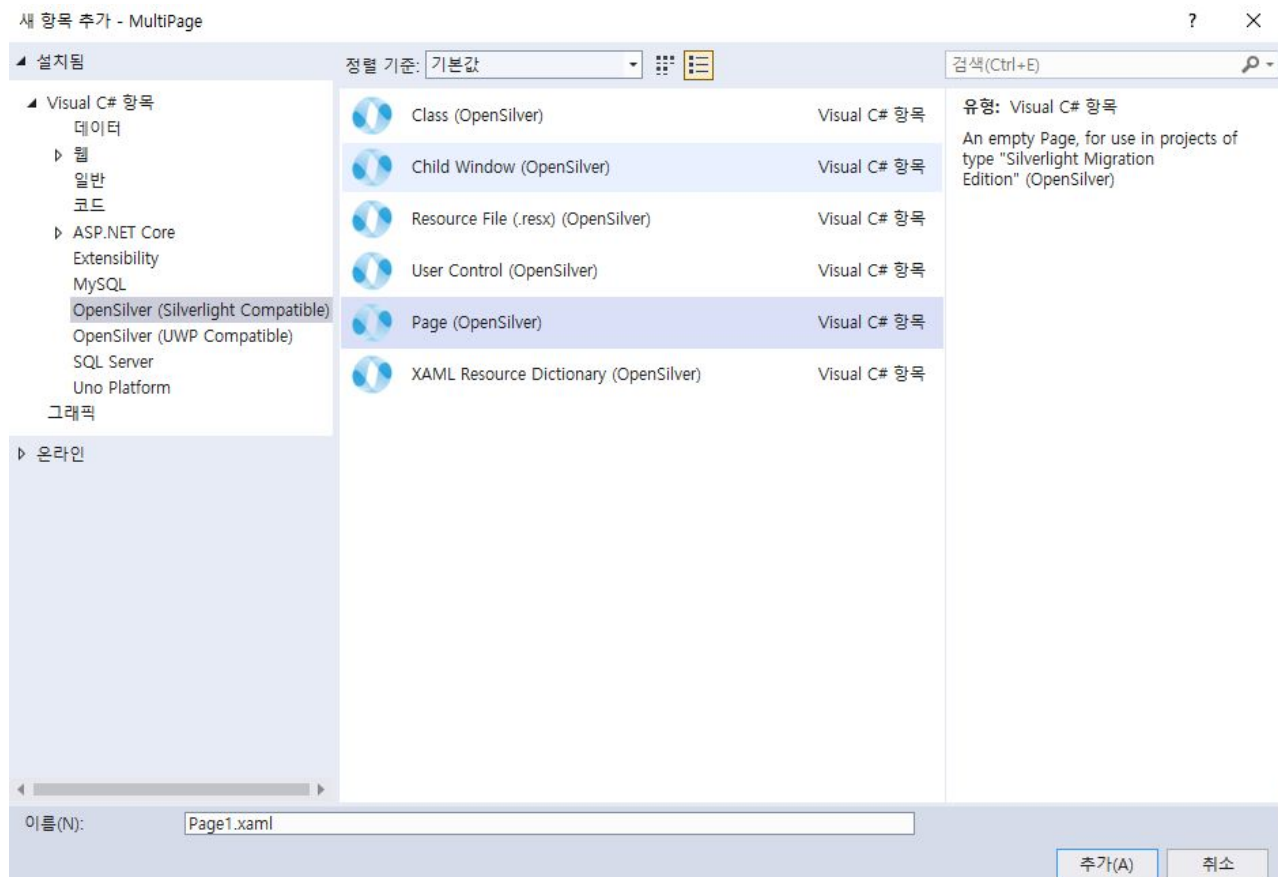
```
1 <Page
2   x:Class="MultiPage.MainPage"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:local="clr-namespace:MultiPage"
6   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8   mc:Ignorable="d">
9   <Frame x:Name="PageContainer" JournalOwnership="Automatic" Margin="5,0,0,30" HorizontalContentAlignment="Stretch">
10    <Canvas>
11      <TextBlock Text="Hello World!" x:Name="TextBlock1" Canvas.Left="20" Canvas.Top="30"/>
12
13      <Button Grid.Row="1" Content="Go Page1" Click="Button_Click" HorizontalAlignment="Center" VerticalAlignment="Center" Canvas.Left="20"
14              Canvas.Top="60"/>
15      <Button Grid.Row="1" Content="Go Page2" Click="Button_Click_1" HorizontalAlignment="Center" VerticalAlignment="Center" Canvas.Left="80"
16              Canvas.Top="60"/>
17    </Canvas>
18  </Frame>
19 </Page>
```

MainPage.xaml.cs MainPage.xaml Page2.xaml Page1.xaml

MultiPage MultiPage.MainPage

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Windows;
6  using System.Windows.Controls;
7
8  namespace MultiPage
9  {
10     참조 4개
11     public partial class MainPage : Page
12     {
13         참조 1개
14         public MainPage()
15         {
16             this.InitializeComponent();
17             // Enter construction logic here...
18         }
19
20         참조 1개
21         private void Button_Click(object sender, RoutedEventArgs e)
22         {
23             Uri uri = new Uri("/Page1.xaml", UriKind.Relative);
24             PageContainer.Source = uri;
25         }
26
27         참조 1개
28         private void Button_Click_1(object sender, RoutedEventArgs e)
29         {
30             Uri uri = new Uri("/Page2.xaml", UriKind.Relative);
31             PageContainer.Source = uri;
32         }
33     }
34 }
```

새로운 Page 추가는 아래처럼 한다



마무리

- **C#**을 알고 있다면 **Blazor, OpenSilver** 프레임워크를 사용하면 웹 프론트엔드 프로그래밍이 기존보다 훨씬 쉬워진다.
- **C#**은 모르고, **HTML+JS**도 잘 모른다면 서버 프로그래밍 특성 상 **C#**을 배우는 것이 훨씬 유리하고 배우기 쉽다. 즉 **Blazor, OpenSilver** 프레임워크를 사용하는 것이 좋다.
- 각 컴포넌트 사용 방법은 예제 코드를 보면 쉽게 이해할 수 있으므로 필요할 때 해당 부분을 보면 된다.
- **WinForm**에 익숙하다면 **Blazor, WPF**에 익숙하면 **OpenSilver**가 배우기 쉽다.
- 개발 측면에서는 **Blazor**이 더 좋은 듯. **Web API** 서버와 연동하기가 훨씬 더 쉽다. 기본으로 프로젝트가 만들어진다.
- **Visual Studio**를 사용할 수 없다면 **Blazor**을 선택해야 한다.
- 이제 웹 프론트엔드 프로그래밍을 무서워하지 말고 **Blazor, OpenSilver**로 자신을 가지고 개발하자!!!