



AvaloniaUI + Raspberry Pi => 전광판 완성!

Windows 와 Raspbian(Raspberry Pi OS)에서 동작하는
UI APP을 Avalonia UI로 만들어 봅니다.

Vincent

현장에서 전광판을 띄우고 싶은데..

- 산업 현장에서 전광판을 띄우고 싶은 경우가 자주 있다.
 - 그런데 고작 전광판 하나를 띄우기 위해, Desktop PC를 준비하고 Windows를 설치하고...너무 번거롭다.
 - 그럴 때 인터넷이 연결된 Raspberry Pi 와 TV하나만 준비하면 전광판을 구성하기 위한 환경은 끝!
-



Avalonia UI

- Avalonia UI는 .NET 크로스 플랫폼 제작에 사용되는 프레임워크로, Pixel-Perfect의 특징을 가지고 있어서 다른 OS에서도 같은 소스, 같은 화면을 보여줄 수 있다는 장점이 있다.
- 여러 .NET 크로스 플랫폼 중 WPF와 가장 유사하여, WPF 개발자들이 쉽게 다른 플랫폼을 경험해볼 수 있다.



개발환경 구성(1)

- Raspberry Pi 구매 :
<https://www.eleparts.co.kr/goods/view?no=9470248>
 - 각자 구매하는 것이 저렴하지만 저 같은 초보자들에게
일단은 생각없이 하드웨어를 구성하여 빨리 Avalonia
UI를 테스트해보고 집중하기에 적합하여 선택.
-



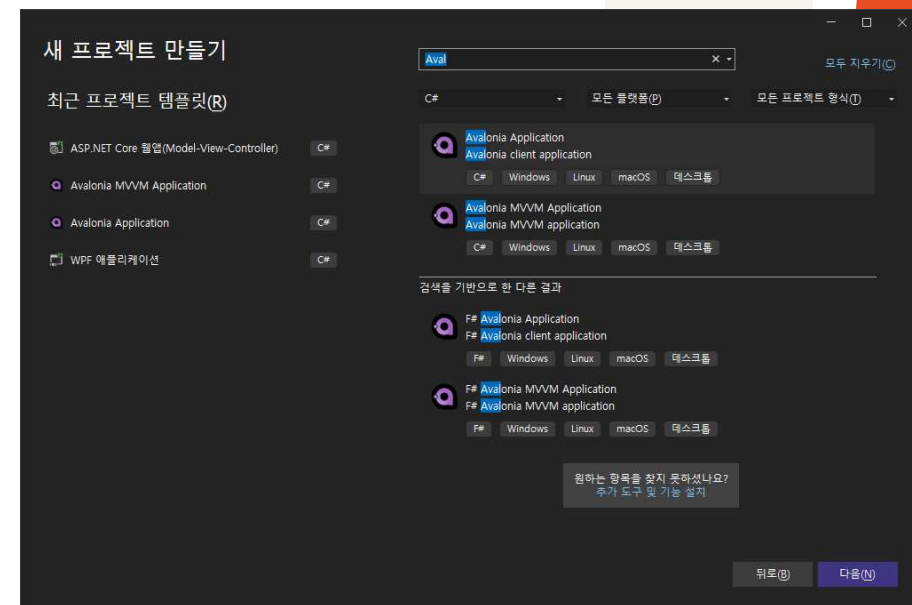
개발환경 구성(2)

- Visual Studio 2022 Community
 - .NET 6
 - Community Toolkit
 - Avalonia UI 0.10.18 (preview-11 가능)
 - <https://marketplace.visualstudio.com/items?itemName=AvaloniaTeam.AvaloniaVS>
 - 위 링크에서 Avalonia UI 용 VS2022 확장을 설치
-



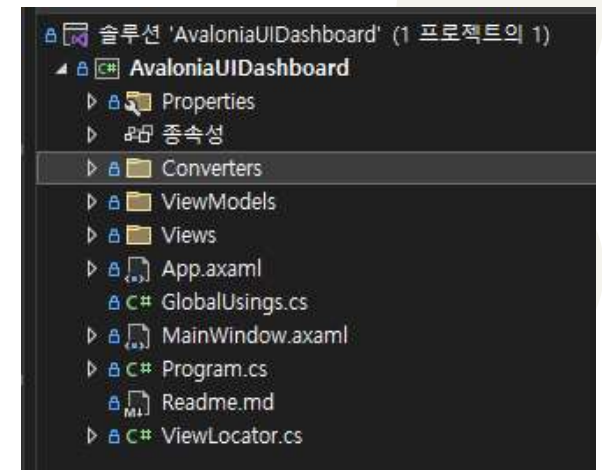
프로젝트 생성

- Avalonia Application 생성
- Avalonia MVVM Application 템플릿은 ReactiveUI를 모르면 사용할 수 없다.



프로젝트 MVVM 구성(1)

- ViewLocator.cs 파일을 생성.
- ViewLocator.cs의 원본은 아래 링크 확인.
- <https://docs.avaloniaui.net/tutorials/todo-list-app/locating-views>
- ViewLocator를 사용하면 이름으로 ViewModel과 View를 매칭하기 때문에 View와 ViewModel 간 의존성 제거.
- 위 링크에 ViewLocator의 메커니즘에 대해서도 설명하고 있습니다.
- Converters, ViewModels, Views 폴더를 생성.



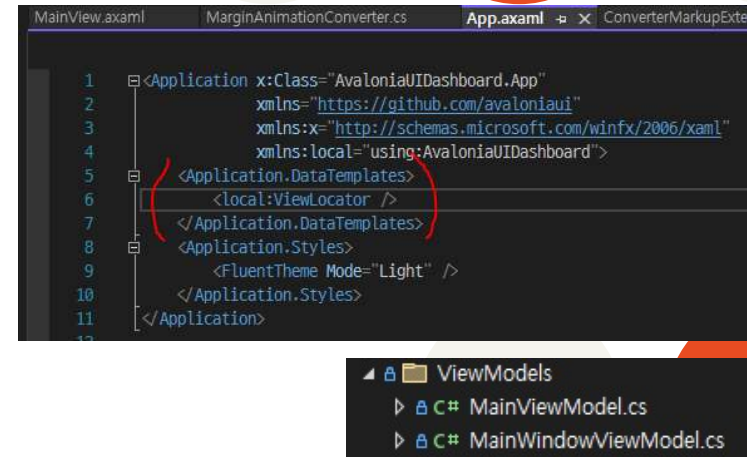
프로젝트 MVVM 구성(2)

- Community Toolkit을 이용하면 MVVM에 필요한 여러 사전작업을 Source Generator를 통해 쉽게 구성 가능.
<https://docs.microsoft.com/ko-kr/dotnet/communitytoolkit/mvvm/>
 - 위 문서에서 사용방법을 자세하게 얻을 수 있으며,
7버전과 8버전에는 API 명칭에 차이가 있으므로, 가급적 8버전으로 작업하는 것을 추천.
-



본격적으로 전광판 개발(1)

- App.axaml에 ViewLocator를 정의한다.
- Axaml에 대한 토론
<https://github.com/AvaloniaUI/Avalonia/issues/4102>
- ViewModels 폴더에 MainViewModel.cs, MainWindowViewModel.cs를 생성한다.



```
1 <Application x:Class="AvaloniaUIDashboard.App"
2           xmlns="https://github.com/avaloniaui"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           xmlns:local="using:AvaloniaUIDashboard">
5   <Application.DataTemplates>
6     <local:ViewLocator />
7   </Application.DataTemplates>
8   <Application.Styles>
9     <FluentTheme Mode="Light" />
10  </Application.Styles>
11 </Application>
```

ViewModels

- ▶ C# MainViewModel.cs
- ▶ C# MainWindowViewModel.cs

본격적으로 전광판 개발(2)

- MainWindowViewModel에 그림과 같이 코딩한다.
- **ObservableObjectAttribute**와 **partial** 키워드로 CommunityToolkit이 소스를 생성한다.
- MainWindowViewModel에는 콘텐츠로 사용될 MainViewModel을 ObservablePropertyAttribute로 정의하고 멤버변수명은 꼭 **소문자**로 시작한다.
- ViewModel만 넣어주면 ViewLocator가 이름으로 매칭하여 View를 위치시켜준다.

```
[ObservableObject]
참조 4개
public partial class MainWindowViewModel
{
    참조 1개
    public MainWindowViewModel()
    {
        MainViewModel = new MainViewModel();
    }

    [ObservableProperty]
    private MainViewModel mainViewModel;
}
```

본격적으로 전광판 개발(3)

- ViewLocator의 코드를 보면 ViewModel 과 View를 키워드로 사용하여 매칭하고 있는데, MainWindow.axaml은 접미사로 View가 붙어있지 않으므로, MainWindow만 따로 DataContext를 MainWindowViewModel로 할당한다.
- 이후 해당 ViewModel이 ObservableObject 타입일 경우 Match 메서드에서 true를 반환하며 View가 그려진다.

```
참조 0개
public override void OnFrameworkInitializationCompleted()
{
    if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime desktop)
    {
        desktop.MainWindow = new MainWindow
        {
            DataContext = new MainWindowViewModel()
        };
    }

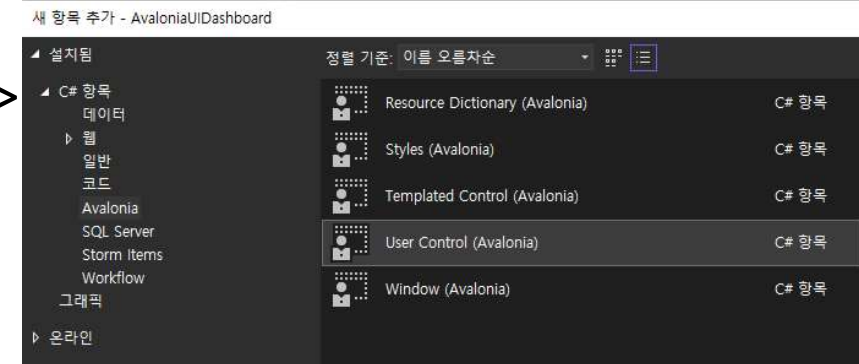
    base.OnFrameworkInitializationCompleted();
}
```

```
string name = data.GetType().FullName!.Replace("ViewModel", "View");
```

```
public bool Match(object data)
{
    return data is ObservableObject;
}
```

본격적으로 전광판 개발(4)

- Views 폴더에 [컨텍스트 메뉴]->[추가]->[새 항목]->[Avalonia]->[UserControl] 을 선택하여, MainView로 명명된 Avalonia UI UserControl을 생성한다.
- MainWindow의 Content 속성에 MainViewModel을 바인딩한다.



```
<Window x:Class="AvaloniaUIDashboard.MainWindow"
        xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:view="clr-namespace:AvaloniaUIDashboard.Views"
        Title="AvaloniaUIDashboard"
        Width="1920"
        Height="1080"
        Content="{Binding MainViewModel}"
        mc:Ignorable="d" />
```

본격적으로 전광판 개발(5)

- WPF는 Resource를 통해 Style을 정의하지만, Avalonia UI에서는 Styles 속성을 통해 Style을 정의한다.
 - WPF의 Animation은 StoryBoard와 DoubleAnimation 등의 클래스와 연계하여 애니메이션을 정의했지만, Avalonia UI에서는 CSS 스타일을 따르므로 KeyFrame를 이용하여 Animation을 정의한다.
 - ViewLocator 사용 시 ObservableObject는 Attribute형태로 적용하면 못 찾으므로, ViewLocator가 View를 찾을 MainView는 상속형태로 정의한다.
-



MainView 개발 및 배포 시연



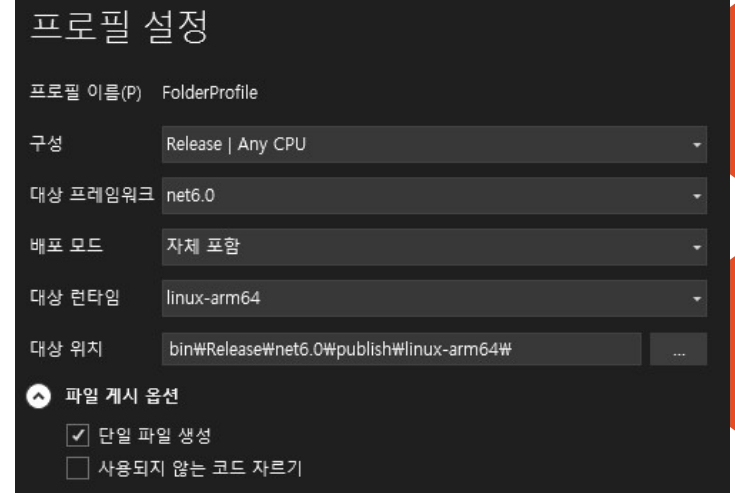
Raspbian 으로 배포

- Visual Studio 2022에서 개발을 끝낸 뒤에는 SCP를 통해 Raspbian으로 .NET Assembly를 전달해야 한다.
- VS 2022의 Publish 기능을 이용하여, 사진과 같은 구성으로 게시를 한 뒤 게시 경로의 모든 파일을 SCP로 전달한다.
- 일반적으로 Windows 10 이상에는 OpenSSH가 설치되어 있으므로 SSH 접속 테스트 후 SCP 명령으로 파일을 복사한다.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\chris> scp -r ./* pi@192.168.219.104:/home/pi/Desktop/AvaloniaUISafetyScoreboard
```



지금까지
'Avalonia UI를 이용한 Raspberry Pi 전광판 개발하기'
의 Vincent 었습니다.

시청해주셔서 감사합니다!!

