.net documentation

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/operators/conditional-operator>

<https://www.wpf-tutorial.com/>

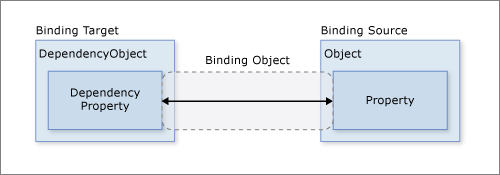
C# 기본 문법 정리하기 – 기본적으로 학습한 것 위주로 정리

1. DataConext

* DataConext 속성은 UI 엘리멘트에 바인딩을 하는 기본 속성. DataContext 는 모든 UI 개체들이 가지고 있으며 최상위 개체에 DataContext 를 설정한 후 자식 개체들에게 바인딩을 할 수도 있으며 별도로 DataConext 를 설정한 후 바인딩 할 수 있다.
* 바인딩 XAML 과 CS 파일에서 두가지 다 가능하다. CS 의 경우는

1. 바인딩

<https://docs.microsoft.com/ko-kr/dotnet/desktop/wpf/data/data-binding-overview?view=netdesktop-5.0>

* 데이터 바인딩은 앱 UI와 해당 UI가 표시하는 데이터를 연결하는 프로세스.
* 
* 바인딩의 방법은 여러 방법이 있는데 부모 요소에서 DataContext 를 정의하고 이를 상속시킬 수 있다. 이러한 방식은 여러속성을 하나의 소스에서 바인딩할 경우 유용하다.
* 또다른 방법으로 DataContext 를 사용하지 않고 바인딩하고자 하는 요소에서 해당 속성에서 바인딩시 Binding.Source 를 활용하여 바인딩 할 수 있다.
* 전체 개체에 대해서 바인딩 하고자 하는 경우는 path 를 생략할 수 있다. ItemsSource="{Binding}"

1. 의존 속성

<https://m.blog.naver.com/PostView.nhn?blogId=jjoommnn&logNo=130034241685&proxyReferer=https:%2F%2Fwww.google.com%2F>

의존 속성에서 의존이라는 단어의 의미는 어떤 객체의 모습 혹은 행동이 해당 의존 속성에 의해서 변경될 수 있음을 의미함.

예를 들어, Button 의 Background Brush 를 바꾸면 Button이 자동적으로 다시 그려지는 것이다. 전통적인 WIN32 프로그램의 경우에서는 Invalidate(), VM-PAINT 를 이용하여 코딩을 통해 해당 개체를 변경할 수 있다.

의존속성은 3가지 구성 요소를 가진다.

* System.Windows.DependencyObject
* System.Windows.DependencyProperty
* System.Windows.PropertyMetadata

DependencyObject 는 의존속성을 가지는 객체이고, WPF의 대부분의 UI 요소들은 DependencyObject 를 상속받아 구현되었다. DependencyObject 는 GetValue, SetValue 메소드를 가지고 있다.

DependencyProperty는 DependencyObject 의 static 멤버이며, DependencyObject에서 SetValue 메소드를 통해 DependencyProperty와 Value 값 등 2개의 인자를 이용해 호출 된다.

PropertyMetadata 는 DependencyProperty 의 부가 정보 정보이며 DependencyProperty에 포함되어 있다.

1/15일 다시 한번 살펴볼 영상

<https://www.aspdotnet-sekhar.com/2013/08/updatesourcetrigger-property-in-wpf.html>

<https://www.wpf-tutorial.com/ko/37/%EB%8D%B0%EC%9D%B4%ED%84%B0-%EB%B0%94%EC%9D%B8%EB%94%A9/updatesourcetrigger-%EC%86%8D%EC%84%B1/>

1. Freezable

<http://okminseok.blogspot.com/2019/04/wpf-binding-proxy.html>

<https://docs.microsoft.com/ko-kr/dotnet/desktop/wpf/advanced/freezable-objects-overview?view=netframeworkdesktop-4.8>

<https://www.python2.net/questions-103833.htm> (참조!)

<https://thomaslevesque.com/2011/03/21/wpf-how-to-bind-to-data-when-the-datacontext-is-not-inherited/>

분석 소스에서는, (일단 지금은 이렇게 추정함) MVVM 을 디자인 패턴을 사용하고 있는데 이때, ViewModel 을 참조하여야 하는 경우가 발생하여(바인딩을 하여야 하는 경우가 발생하여) BindingProxy 를 생성하여 이를 통해서 바인딩한 것으로 추정됨.

Freezable 객체의 경우, Visual Tree 또는 Logical Tree가 다른 경우에도 DataContext를 상속할 수 있도록 하는 기능을 가지고 있다.

1. .net 에서 람다식

* 람다식은 익명함수로 인식하면 간단하다.
* 람다식으로 작성하면 코드가 간단해진다는 장점이 있다.
* ‘=>’ 연산자를 기준으로 좌변은 입력 파라미터, 우변은 표현식이 작성된다. (파라미터 => 표현식 또는 파라미터 => 처리구문)

|  |
| --- |
| x => x\*10  (int x) => x\*10  (int x) => (x\*10)  x => {return x\*10;}  (int x) => {return x\*10;} |

* 파라미터의 타입을 선언해도 되고 생략해도 되고, 타입선언을 하면 괄호를 쳐야하고, return를 써도되고 안써도되는데 쓰려면 {}로 감싸줘야 한다.
* 파라미터가 2개 이상일 때 아래와 같이 처리한다.

|  |
| --- |
| (x, y) => x\*y  (int x) => x\*10  (int x, int y) => x\*y |

보강할 내용

<https://m.blog.naver.com/PostView.nhn?blogId=bug_ping&logNo=221432802633&proxyReferer=https:%2F%2Fwww.google.com%2F>

1. net Framework 와 .net core 에서의 WPF 차이점.

* .net core 용 WPF 는 원래 .net framework 용 젤 소스코드에서 분기된 오픈 소스 프레임워크 임.
* .net framework 기능 중에 .net core 에서 지원하지 않는 몇가지 기능들이 있음.
* .net core 는 sdk 스타일 프로젝트 파일을 사용함.
  1. 각각 의 장점

.net framework

- 기존 .net framework 사용자의 경우 빠르게 프로젝트 진행가능

- .net framework 의 업그레이드가 필요 없는 경우.

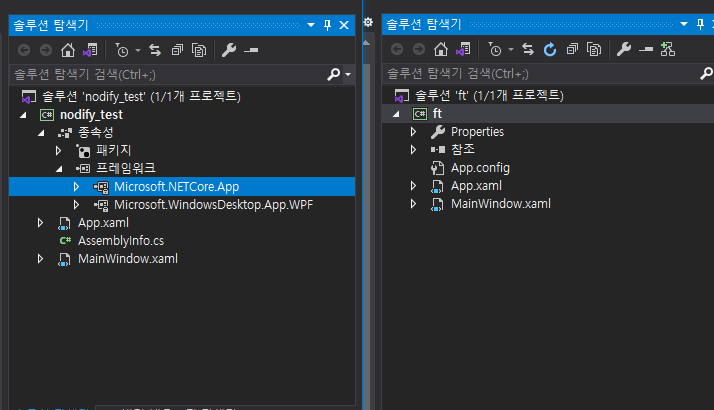
-.net core 의 장점

- 지속적인 프레임워크의 업데이트가 필요한 경우.

- ms 에서는 .net core 를 중심으로 향후 업데이트 예상됨.

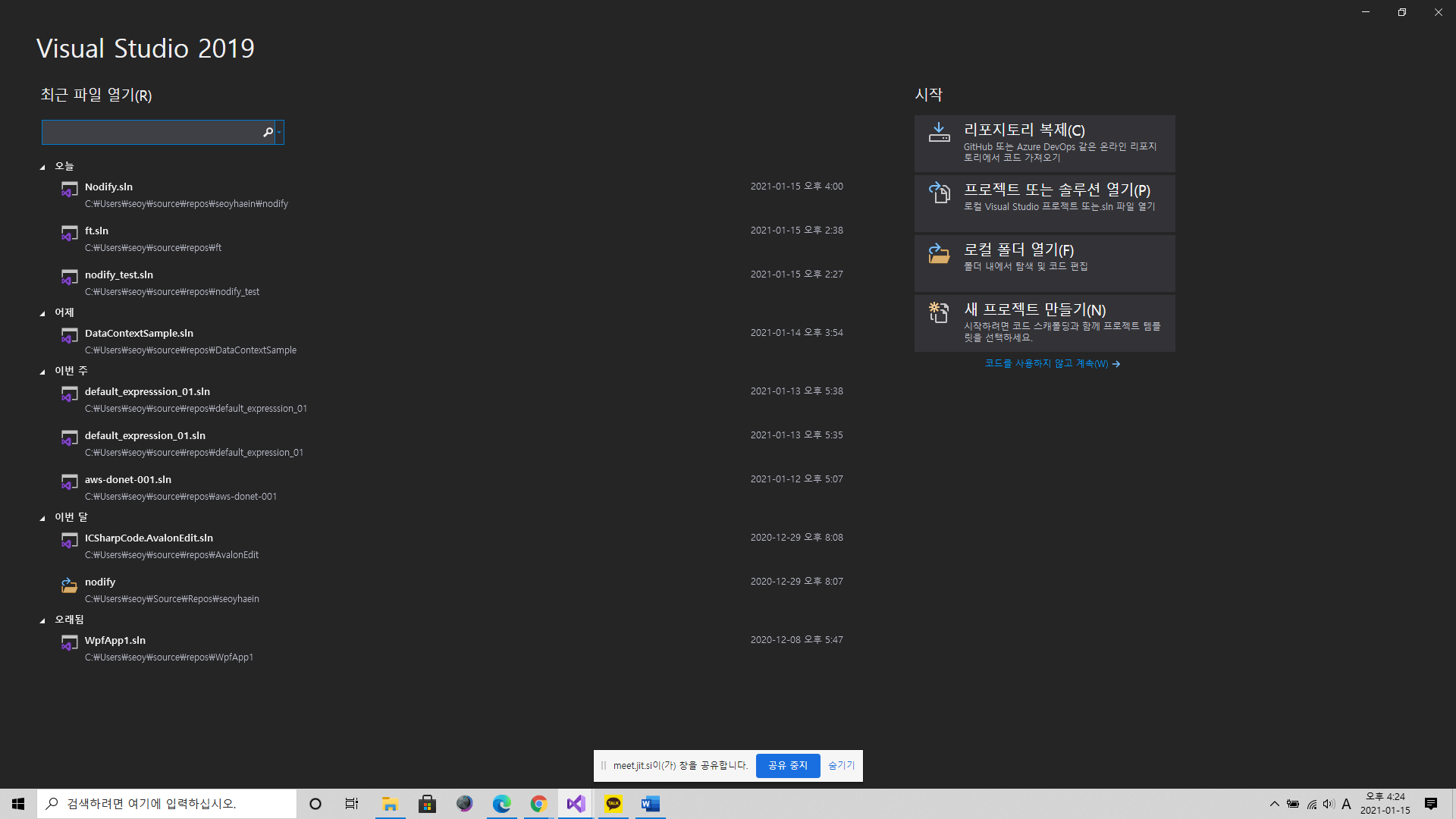
- .net framework 은 윈도우 및 웹 만 가능했지만 .net core 의 경우는 윈도우, 리눅스, macos 에서 사용 가능

- docker 사용 가능, 경량화로 인해 매우 가벼움.

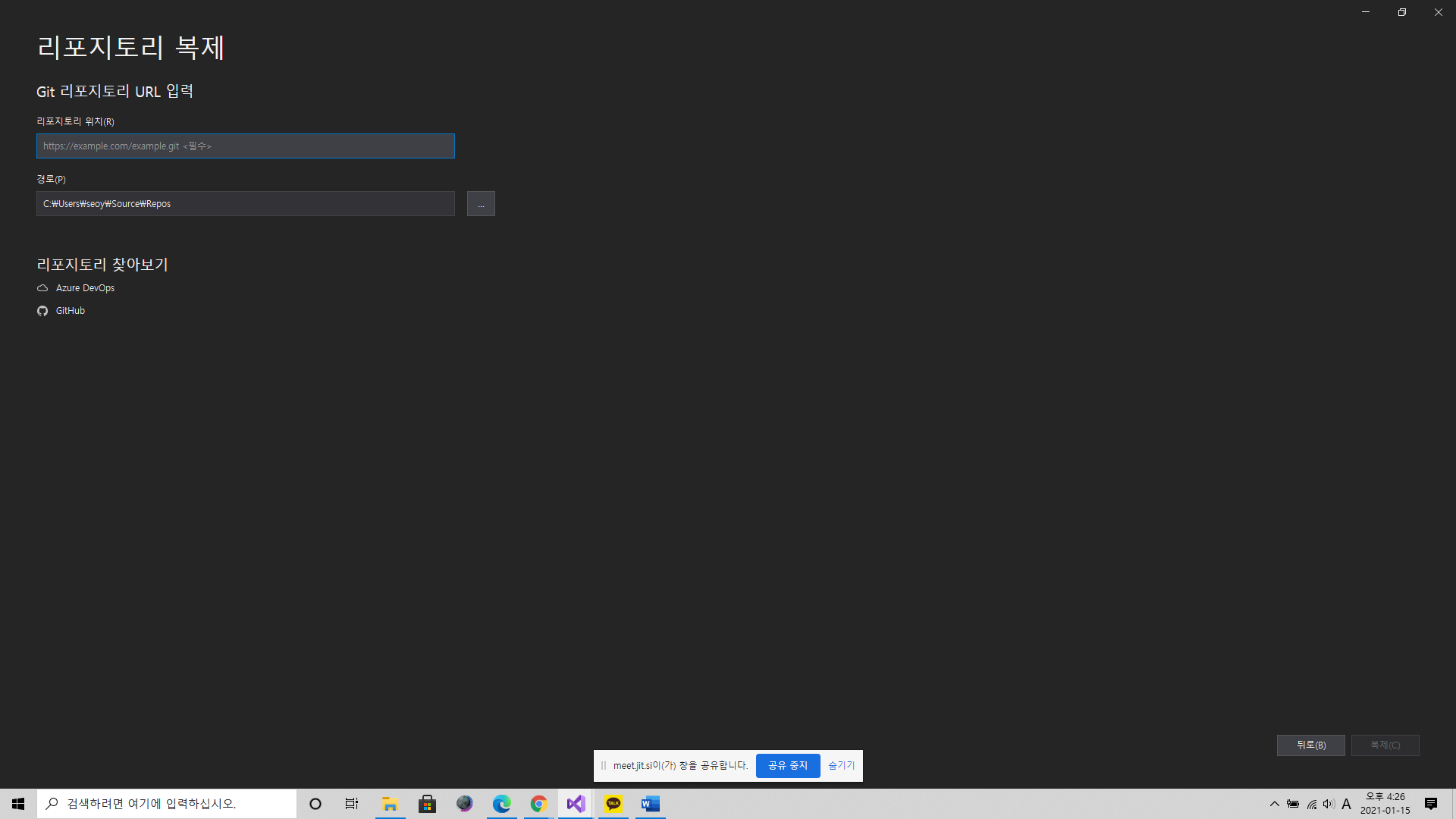


<그림1 .net core wpf> <그림2 .net framework wpf>

1. Visual Studio 2019 에서 github 사용하기



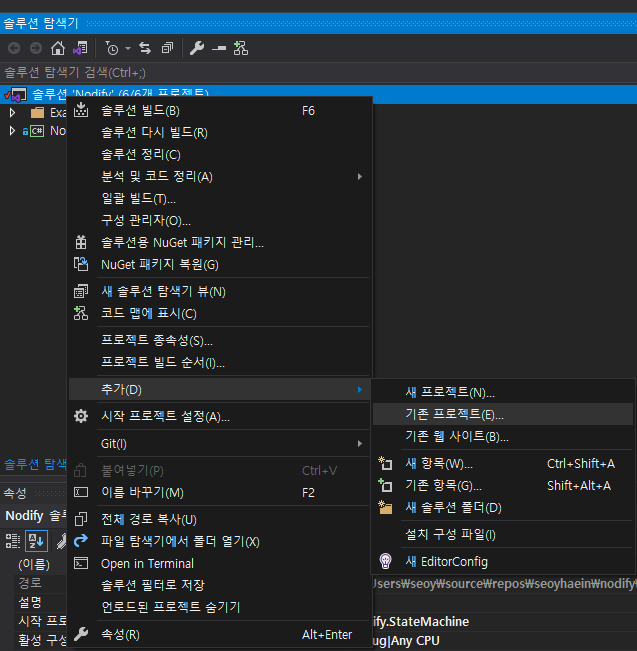
레포지토리 복제를 클릭하여 github 주소를 입력한다.



잠시 보류

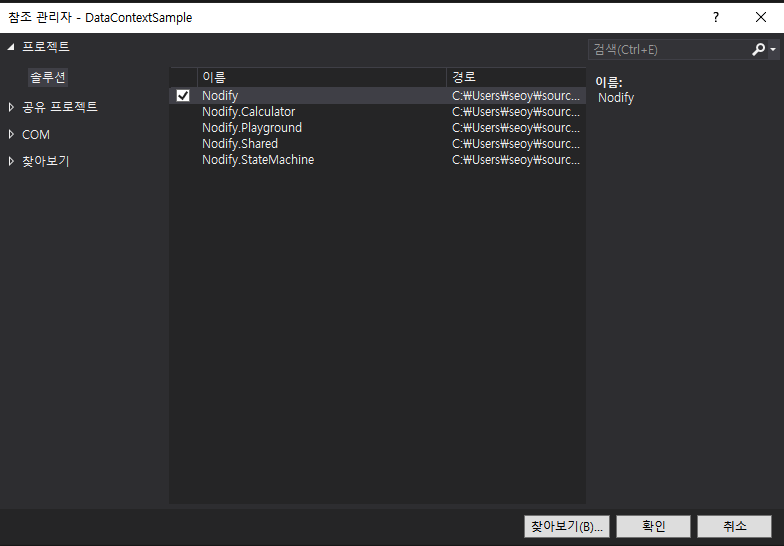
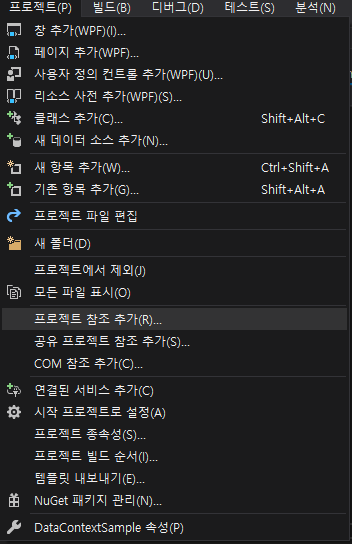
<https://smoh.tistory.com/269>

1. 솔루션에 프로젝트 추가하기



솔루션 오른쪽 마우스 클릭 -> 추가 -> 기존 프로젝트 또는 새프로젝트

* 1. 추가된 프로젝트에서 해당 솔루션의 프로젝트를 종속성 추가하기



1. XAML – xmlns 기억 되짚어 보기

* XAML 파일에서 최상위 태그인 <Window> 태그 내에서 x:Class=, xmlns=, xmlns x:=, … 등이 있다. 여기서 x:Class= 의 경우는 해당 프로젝트의 네임스페이스라고 이해하면 쉬울 수 있으며, xmlns= 는 기본적으로 cs 파일에서 using 과 같은 의미를 가지고 있다. xmlns 는 사실 XAML도 xml 에서 파생되었음으로 특정 xaml 태그가 중복되는 것을 방지 위한 네임스페이스 개념이다. 따라서, 해당 키를 기준으로 구분하여 서로 동일한 이름의 태그라고 하더라도 해당 키를 기준으로 태그를 사용할 경우, 충돌이 발생하는 것을 막을 수 있다. 만약 xmlns 에서 키가 설정이 되어 있지 않다면 WPF 기본 UI 요소에 대한 태그를 정의하고 있다. 그리고 x:= 와 같이 키를 설정할 수 있다.
* 사용자 정의 컨트롤을 사용하는 경우는 AssemblyInfo.cs 파일에서 아래와 같이 정의 하고 있으며

|  |
| --- |
| [assembly: XmlnsDefinition("http://miroiu.github.io/winfx/xaml/nodify", "Nodify")]  [assembly: XmlnsPrefix("http://miroiu.github.io/winfx/xaml/nodify", "nodify")] |

위와 같이 작성된 경우는,

xmlns:nodify="http://miroiu.github.io/winfx/xaml/nodify

과 같이 xaml 에서 선언해서 사용할 수 있다.

1. 도구상자에 xaml 코드 넣기

<https://happybono.wordpress.com/2017/07/25/visual-studio-%EC%9D%98-%EC%88%A8%EA%B2%A8%EC%A7%84-%EA%B8%B0%EB%8A%A5-1-toolbox-%EB%8F%84%EA%B5%AC-%EC%83%81%EC%9E%90-%EC%97%90-%EC%86%8C%EC%8A%A4-%EC%BD%94%EB%93%9C-%EC%A1%B0%EA%B0%81-%EB%B3%B4/>

1. Nuget 에서 사용자 정의 컨트롤 추가할 때 도구상자에 나타나지 않는데 이 문제는 어떻게 해결하지??
2. DependencyProperty.Register

<https://www.youtube.com/watch?v=tsLiFNakLrc>

<https://github.com/SingletonSean/WPFCustomControls>

<https://docs.microsoft.com/ko-kr/dotnet/api/system.windows.dependencyproperty.register?view=net-5.0>

[FrameworkPropertyMetadata Class (System.Windows) | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/api/system.windows.frameworkpropertymetadata?view=net-5.0)

<https://wpf.2000things.com/2012/06/07/575-propertymetadata-vs-frameworkpropertymetadata/>

1. Action<T>delegate, Func<T, TResult>delegate, …

<http://www.csharpstudy.com/Tip/Tip-Func.aspx>

Action<T> 인경우 T 는 입력값이 T type 이다.

1. ObservableObject

<https://zinzza.tistory.com/16>

코드에서는 MVVM 을 구현하기 위해서 ObserableObject 를 구현해놓았다.

1. MultiSelector 를 통해서 구현한 사례

<https://www.codeproject.com/Articles/182683/NetworkView-A-WPF-custom-control-for-visualizing-a>

1. 식 본문 멤버

<https://docs.microsoft.com/ko-kr/dotnet/csharp/programming-guide/statements-expressions-operators/expression-bodied-members>

* 식 본문 멤버를 사용하여 메서드, 일기 전용 속성, Property, 생성자, 종료자, 인덱서 등을 간결하게 정의하여 사용할 수 있다.
* Member => expression; 으로 표현할 수 있다. Expression 은 단일식이어야 한다.

1. 이벤트 생성 및 처리

* 사용자 지정 라우트 된 이벤트 만들기
* 이벤트 발생시키기
* 이벤트 핸들러 추가하기
* 이벤트 만들어 주기

1. 정적 생성자

<https://docs.microsoft.com/ko-kr/dotnet/csharp/programming-guide/classes-and-structs/static-constructors>

18.1정적 클래스

<https://docs.microsoft.com/ko-kr/dotnet/csharp/programming-guide/classes-and-structs/static-constructors>

1. ItemControl

<https://m.blog.naver.com/PostView.nhn?blogId=jjoommnn&logNo=130131363921&proxyReferer=https:%2F%2Fwww.google.com%2F>

1. Attached Property

<https://kaki104.tistory.com/562>

1. Template binding

<https://mhchoi8423.tistory.com/1>

1. sealed 한정자

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/sealed>

1. 연산자

<https://velog.io/@jinuku/C-%EB%B0%8F-.-%EC%97%B0%EC%82%B0%EC%9E%90>

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/operators/conditional-operator>

Nullable

<https://blockdmask.tistory.com/360>

1. Attribute

<https://qzqz.tistory.com/246>

<https://stackoverflow.com/questions/2968597/what-is-brackets-in-net>

<https://www.codeproject.com/Articles/2933/Attributes-in-C>

1. Params

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/params>

1. DataTemplate 좀더 파악해야함.

<https://charlesk.tistory.com/78>

1. S3

<https://docs.ceph.com/en/latest/radosgw/s3/csharp/>

1. 참조로 전달(학습해야함.)

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/ref>

1. 확장 메소드

<https://devhoma.tistory.com/71>

1. [CallerMemberName] 외 기타

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/attributes/caller-information>

1. Event

<https://www.twilio.com/blog/understanding-wpf-routed-events-in-dot-net-core>

1. Delegate

<https://docs.microsoft.com/ko-kr/dotnet/csharp/programming-guide/delegates/how-to-declare-instantiate-and-use-a-delegate>

1. Internal 그리고 static

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/internal>

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/accessibility-levels>

* Static 을 쓰면 new 를 할필요없이 PendingConnection.GetPotentialConnector 이런식으로 호출 할 수 있다. 즉 클래스이름.함수이름() 이런식으로 호출할 수 있다.

1. Where 제약조건

[where(제네릭 형식 제약 조건) - C# 참조 | Microsoft Docs](https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/where-generic-type-constraint)

1. protected override Size ArrangeOverride(Size arrangeSize) 와 protected override Size MeasureOverride(Size constraint)

<https://ikeptwalking.com/wpf-measureoverride-arrangeoverride-explained/>