



Enlace de Datos

Ricardo Vasquez Sierra

www.ricardovasquez.postach.io



Microsoft
CERTIFIED
Professional

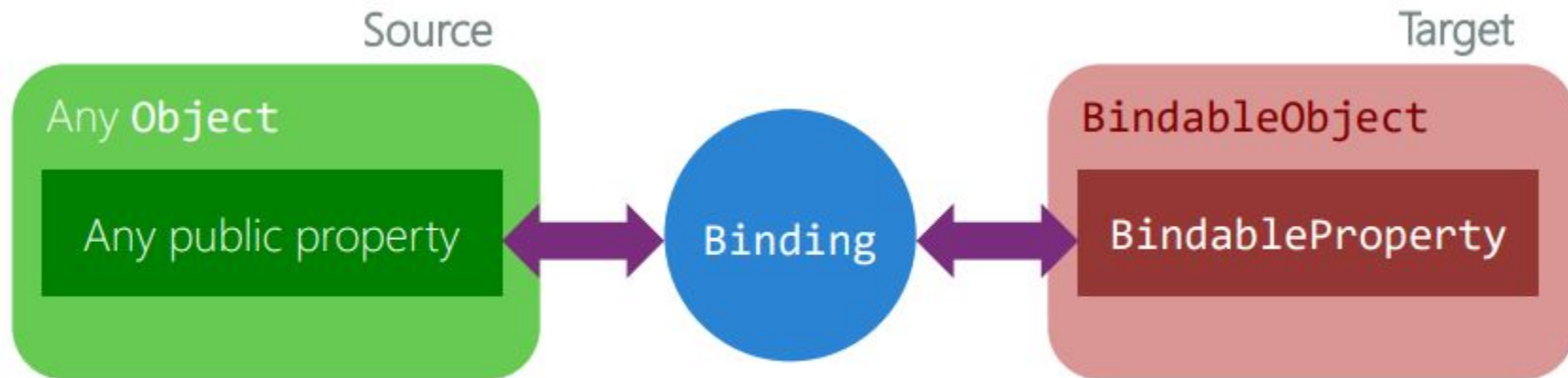
Agenda

- ❖ Enlazar datos a UI
- ❖ Crear enlaces en código
- ❖ Trabajar con contexto de enlace (BindingContext)
- ❖ Cambiar modos de enlace
- ❖ Implementar notificaciones property changed



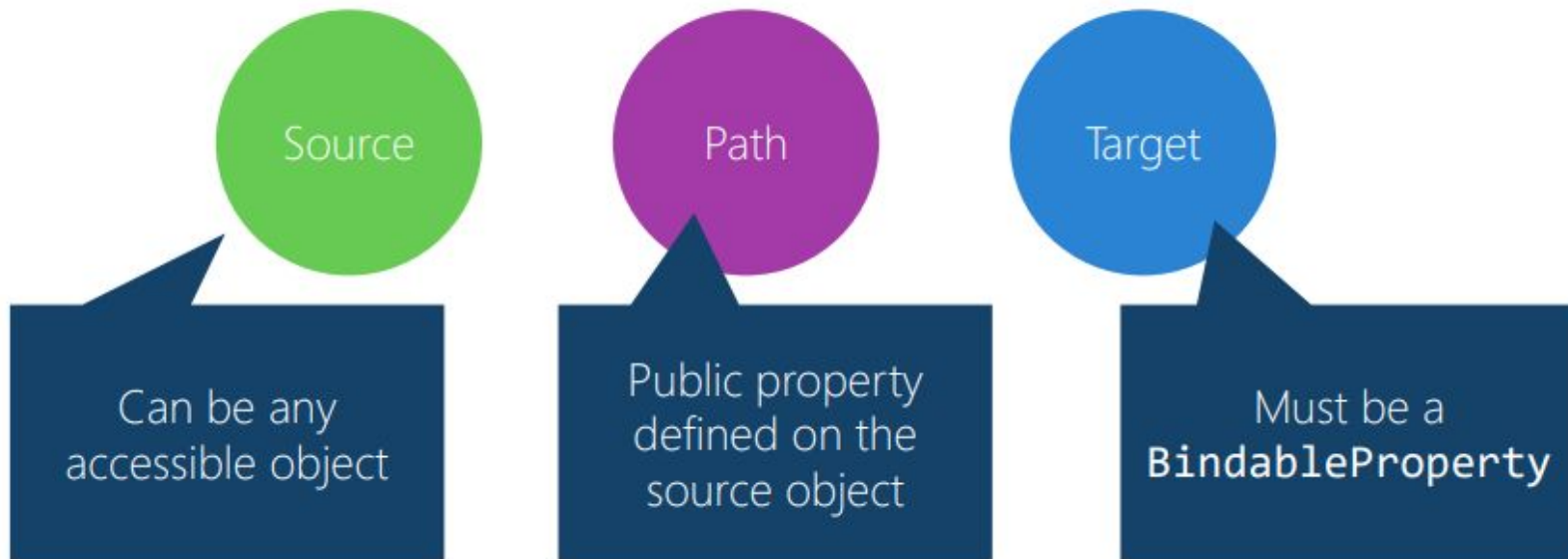
Enlace de Datos (BindingContext)

- ❖ Involucra crear una relación desacoplada entre una **propiedad origen (source)** y una **propiedad destino (target)**, de manera que ambas partes desconozcan la existencia de la otra.



Enlace de Datos

- ❖ Los enlaces de datos constan de 3 partes:



Creando Enlaces de Datos (Source)

```
Todo task = new Todo() { Title = "Pickup some Milk", ... };  
  
Entry Name = new Entry();  
  
Binding nameBinding = new Binding();  
nameBinding.Source = task;  
  
...
```

Binding identifies the **source** of the binding data – this is where the data comes from, in this case it's a single **ToDo** object defined in our application



Creando Enlaces de Datos (Path)

```
Todo task = new Todo() { Title = "Pickup some Milk", ... };  
  
Entry Name = new Entry();  
  
Binding nameBinding = new Binding();  
nameBinding.Source = task;  
nameBinding.Path = "Title";  
  
...
```

Binding identifies the *property path* which identifies a property on the source to get the data from, in this case we want to get the value from the **Todo.Title** property



Creando Enlaces de Datos (Path)

```
Todo task = new Todo() { Title = "Pickup some Milk", ... };
```

```
Entry Name =
```

```
Binding nameB  
nameBinding.S  
nameBinding.P
```

More Path Examples

```
new Binding("Property")  
new Binding("Property.Child")  
new Binding("Property[Key]")  
new Binding("Property[1]")  
new Binding("[Key]")  
new Binding(".")
```



Creando Enlaces de Datos (Target)

```
Todo task = new Todo() { Title = "Pickup some Milk", ... };  
  
Entry Name = new Entry();  
  
Binding nameBinding = new Binding();  
nameBinding.Source = task;  
nameBinding.Path = "Title";  
  
Name.SetBinding(Entry.TextProperty, nameBinding);
```

This is passed the specific target property the binding will work with – this must be a **BindableProperty**



Creando Enlaces de Datos (Target)

```
Todo task = new Todo() { Title = "Pickup some Milk", ... };  
  
Entry Name = new Entry();  
  
Binding nameBinding = new Binding();  
nameBinding.Source = task;  
nameBinding.Path = "Title";  
  
Name.SetBinding(Entry.TextProperty, nameBinding);
```

... and the binding which identifies the source and the property on the source to apply



Creando Enlaces de Datos en XAML

- ❖ Pueden crearse mediante el uso de la extensión de marcado **{Binding}**

```
<StackLayout Padding="20" Spacing="20">
  <StackLayout.Resources>
    <ResourceDictionary>
      <Todo x:Key="getMilk" Title="Pickup some Milk" />
    </ResourceDictionary>
  </StackLayout.Resources>
  <Entry Text="{Binding Title,
    Source={StaticResource getMilk}}" />
  ...
</StackLayout>
```

Assigned to Target property

{Binding} takes the Path as the first unnamed argument

Source supplied through resource

Multiples Enlaces de Datos

- ❖ La propiedad **BindingContext** proporciona el origen para cualquier enlace asociado a una vista cuando la propiedad **Binding.Source** NO ha sido establecida.

```
Todo task = new Todo() { Title = "Buy a Surface Studio", ... };  
...  
Name.BindingContext = task;  
Name.SetBinding<Todo>(Entry.TextProperty,  
                      data => data.Title, BindingMode.TwoWay);
```

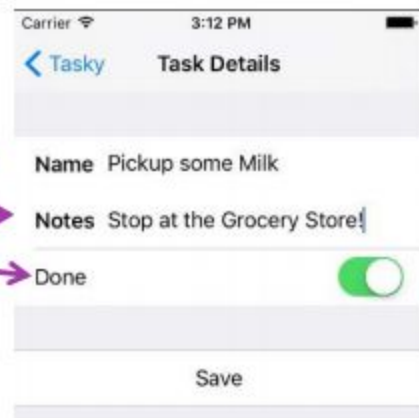


Useful to use a generic form of **SetBinding** to create bindings with typed properties when establishing bindings in code, notice we are *not* setting a source property on the binding – instead, it will use **BindingContext**

Herencia de Contexto de Enlace

- ❖ La propiedad **BindingContext** se hereda de padre a hijo, por lo que puede ser establecida una sola vez en el constructor de la vista raíz y esta será utilizada por todos los componentes de la misma.

```
public partial class TaskyDetailsPage : ContentPage
{
    public TaskyDetailsPage (Todo task)
    {
        BindingContext = task;
        InitializeComponent ();
    }
}
```




Herencia de Contexto de Enlace

- ❖ La propiedad **BindingContext** se hereda de padre a hijo, por lo que puede ser establecida una sola vez en el constructor de la vista raíz y esta será utilizada por todos los componentes de la misma.

```
BindingContext = new Todo() { Title = "Buy a Surface Studio" };
```

```
<StackLayout Padding="20" Spacing="20">  
    <Entry Text="{Binding Title}" />  
    <Entry Text="{Binding Notes}" />  
    <Switch IsToggled="{Binding Completed}" />  
</StackLayout>
```



By setting the binding context to the **Todo**, no explicit source is necessary in XAML

Demostración

Utilizando Enlace de Datos



Enlaces Vista-a-Vista

- ❖ **{x:Reference}** identifica elementos con nombre definidos dentro de la misma página XAML y puede ser utilizado como **source** para un enlace de datos.

```
<StackLayout Padding="20" Spacing="20">
    <Label Text="Hello, Bindings" TextColor="Blue" ...
        Rotation="{Binding Source={x:Reference slider},
            Path=Value}" />
    ...
    <Slider x:Name="slider" Minimum="0" Maximum="360" />
</StackLayout>
```



Modos de Enlace

- ❖ La propiedad **Mode** controla la dirección de los datos transferidos. Puede ser configurado como “**TwoWay**” para habilitar enlaces bidireccionales.

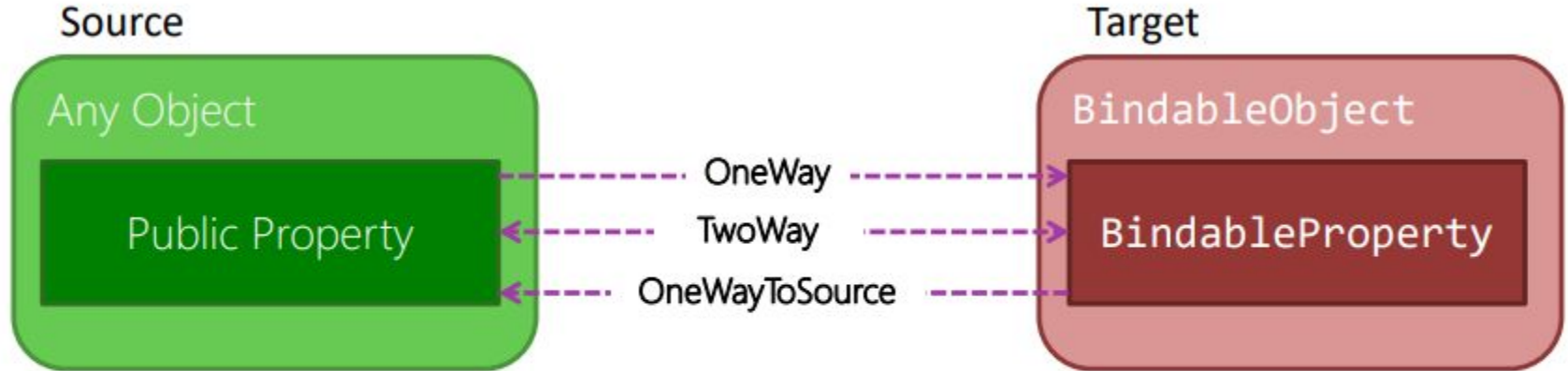
```
Notes.SetBinding(Entry.TextProperty,  
    new Binding("Notes") {  
        Mode = BindingMode.TwoWay  
    });
```

```
<Entry  
    Text="{Binding Notes, Mode=TwoWay}" />
```

Source Property must
have **public setter**

Manually controlled through the
Binding.Mode property

Modos de Enlace Disponibles



Actualizando UI

- ❖ **INotifyPropertyChanged** provee un contrato de notificación de cambios y debe de ser implementado por cualquier modelo modificable al cual se esté creando un enlace.

```
namespace System.ComponentModel
{
    public interface INotifyPropertyChanged
    {
        event PropertyChangedEventHandler PropertyChanged;
    }
}
```


Implementando INotifyPropertyChanged

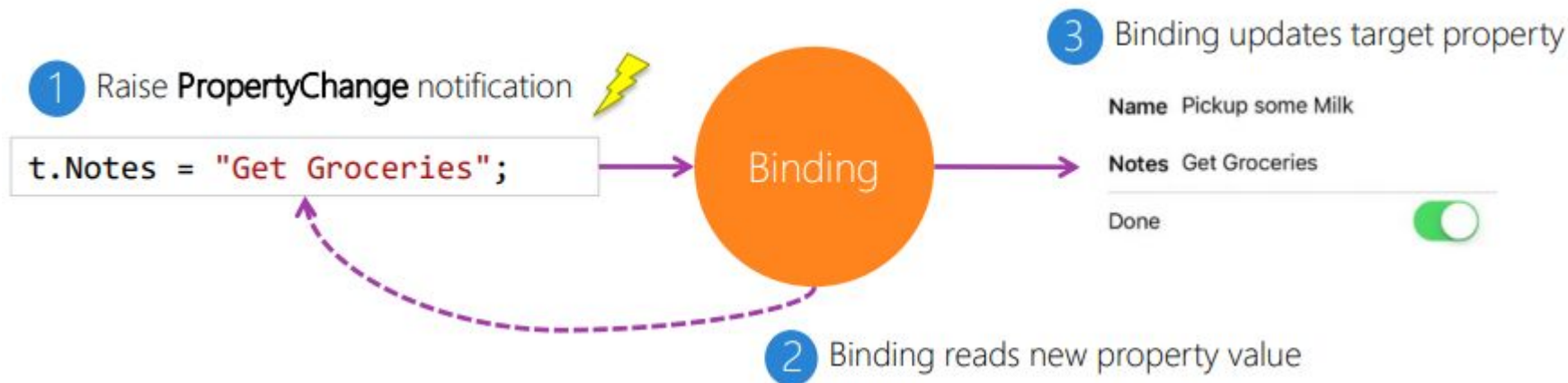
```
public class Todo : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    string notes;
    public string Notes {
        get { return notes; }
        set {
            if (notes != value) {
                notes = value;
                PropertyChanged?.Invoke(
                    this, new PropertyChangedEventArgs(nameof(Notes)));
            }
        }
    }
}
```

Must raise the **PropertyChanged** event when any property is changed – otherwise the UI will not update

INPC + Enlaces de datos

- ❖ Los enlaces de datos estarán suscritos al evento **PropertyChanged** y actualizarán la propiedad **target** cuando detecten la notificación de la propiedad **source**.



Demostración

Utilizando INotifyPropertyChanged





Gracias! :)

Referencias:

Xamarin University



Microsoft
CERTIFIED
Professional