Xamarin Evolve 2014

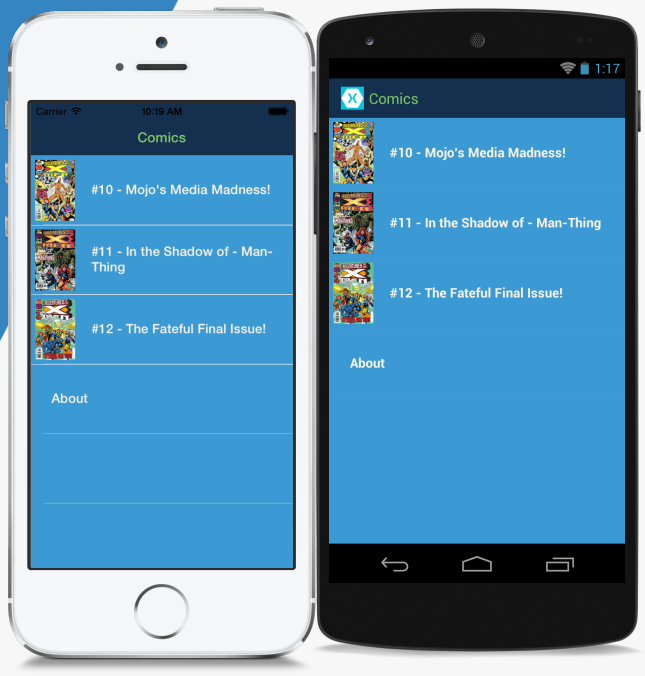
Data Binding in Xamarin.Forms: not your dad's binding

Chris van Wyk
chris.vanwyk@xamarin.com

Xamarin University

Agenda

1. Mapping Data to Visuals
2. Creating Binding in
   a. Code
   b. XAML
3. The Binding Context
4. Property Change Notifications



Xamarin University

Mapping Data to Visuals

Xamarin University

Apps are driven by data

Most applications display and manipulate data in some form

- Cloud-based
- Internally generated

Classes created to represent data are often referred to as Models

- can also refer to "entity" objects

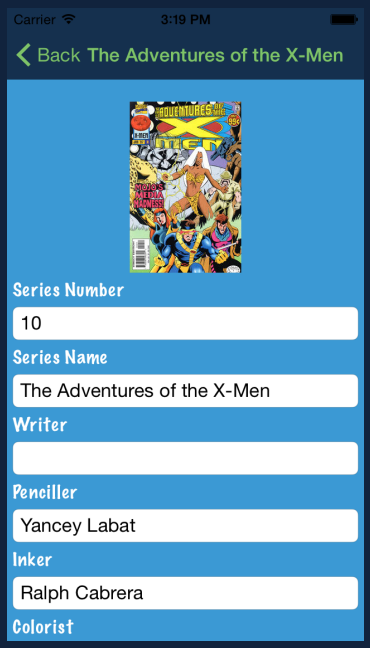
Xamarin University

Models > Visuals

Code maps data from models to the screen

```
SeriesNameLbl = new Label {
  Text = Language.SeriesName,
  ...
} ;

SeriesNameEntry = new Entry{
  Text = _comicBook.SeriesName
} ;
```

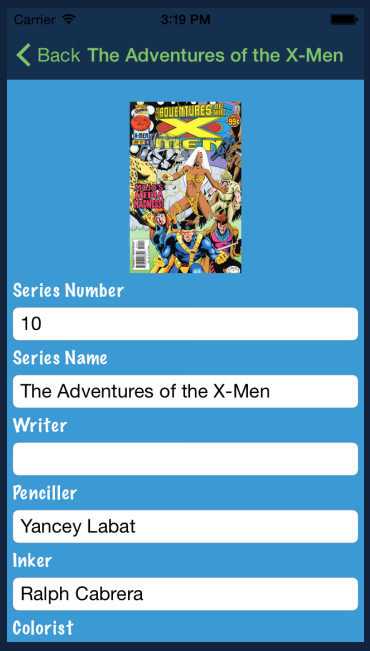


Xamarin University

Models > Visuals

...and provide interactivity / behavior

```
SeriesNameEntry.TextChanged += (sender, e) => {
      if (!e.NewTextValue
        .Equals(_comicBook.SeriesName)){
          _comicBook.SeriesName = e.NewTextValue;
      }
} ;
```



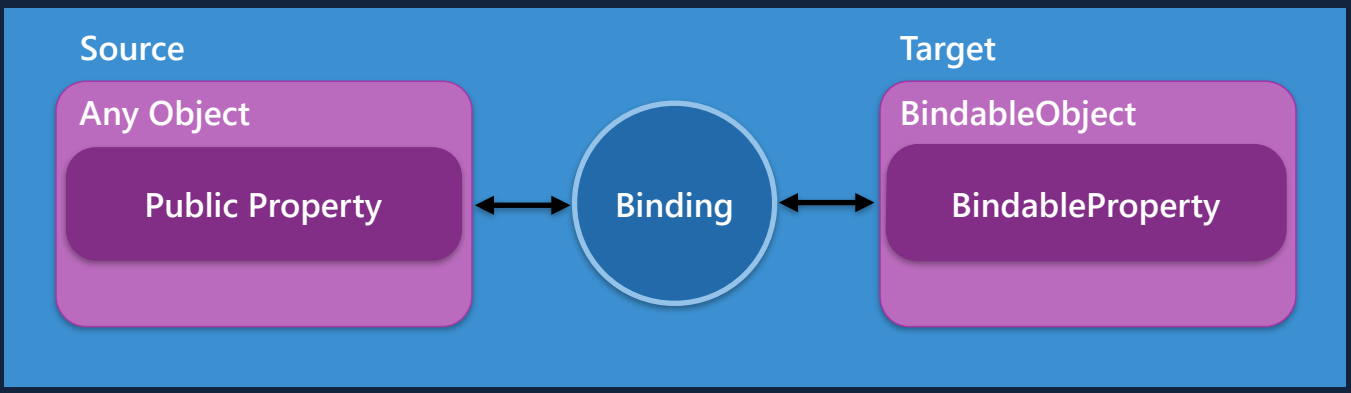
Xamarin University

Data Binding

Xamarin University

Data Binding

Data Binding involves creating a loose relationship between a source property and a target property, source and target are unaware of each other.

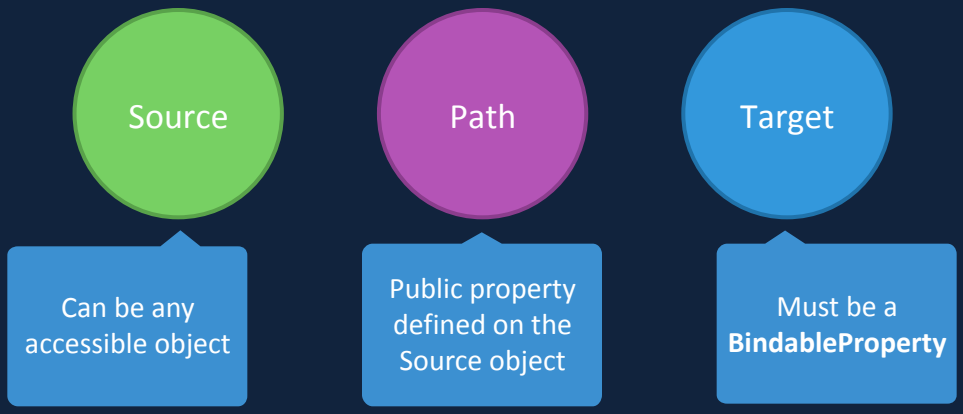


Binding acts as an *intermediary* – moving the data to and from the source and target.

Xamarin University

Creating Bindings

Bindings require three pieces of information



Xamarin University

Creating bindings [source]

```
void ExampleBindingSetup (){
    ComicBook comicBook = new ComicBook() {SeriesName = "The Avengers"};

    Entry seriesNameEntry = new Entry();

    seriesNameEntry.BindingContext = comicBook;

    Binding seriesNameB

    seriesNameEntry.Set                                    Binding);
}
```

ComicBook is the ***data source*** and is set as the **BindingContext** for the entry, all bindings applied to the control will look here for the data

Xamarin University

Creating bindings [source]

```
void ExampleBindingSetup (){
    ComicBook comicBook = new ComicBook() {SeriesName = "The Avengers"};

    Entry seriesNameEntry = new Entry();

    seriesNameEntry.BindingContext = comicBook;

    Binding seriesNameBinding = new Binding("SeriesName");

    seriesNameEntry.SetBinding(Entry.TextProperty, seriesNameBinding);
}
```

Binding object defines the ***property path*** to get the source data from, can use "." for overall source object

Xamarin University

Creating bindings [source]

```
void ExampleBindingSetup (){
    ComicBook comicBook = new ComicBook() {SeriesName = "The Avengers"};

    Entry seriesNameEntry = new Entry();

    seriesNameEntry.BindingContext = comicBook;

    Binding seriesNameBinding = new Binding("SeriesName");

    seriesNameEntry.SetBinding(Entry.TextProperty, seriesNameBinding);
}
```

Binding is associated to the target property using the **BindableObject.SetBinding** method

Xamarin University

Creating bindings [XAML]

Create bindings in XAML with {Binding} markup extension

```
<StackLayout Padding="20" Spacing="20">

    <StackLayout.Resources>
        <ResourceDictionary>
            <ComicBook x:Key="avengers" Name="The Avengers" ... />
        </ResourceDictionary>
    </StackLayout.Resources>

    <Entry  BindingContext="{StaticResource avengers}"
            Text="{Binding Name}" />
    ...
</Sta
```

Source

Target property

Path as 1st unnamed target

Xamarin University

Demo

Xamarin University

The Binding Context

Xamarin University

Binding to model objects

- Source for a binding can be any public property on any object
- Common to use model object to represent data for UI
- Data is then mapped from the model to the UI using bindings

Xamarin University

Utilizing BindingContext

- Many screens pull data from a singular object – mapping different properties to UI controls
- BindingContext would be the same for every control on the page in this case

Xamarin University

BindingContext inheritance

BindingContext is automatically inherited from parent to child

```
public partial class XamlBinding : ContentPage
{
     public XamlBinding (ComicBookViewModel comicBookVM)
     {
          BindingContext = comicBookVM;

          InitializeComponent ();
     }
}
```

Can set the BindingContext ***once*** on the root page and it is then ***assigned*** to ***every control*** (child) on that page

Xamarin University

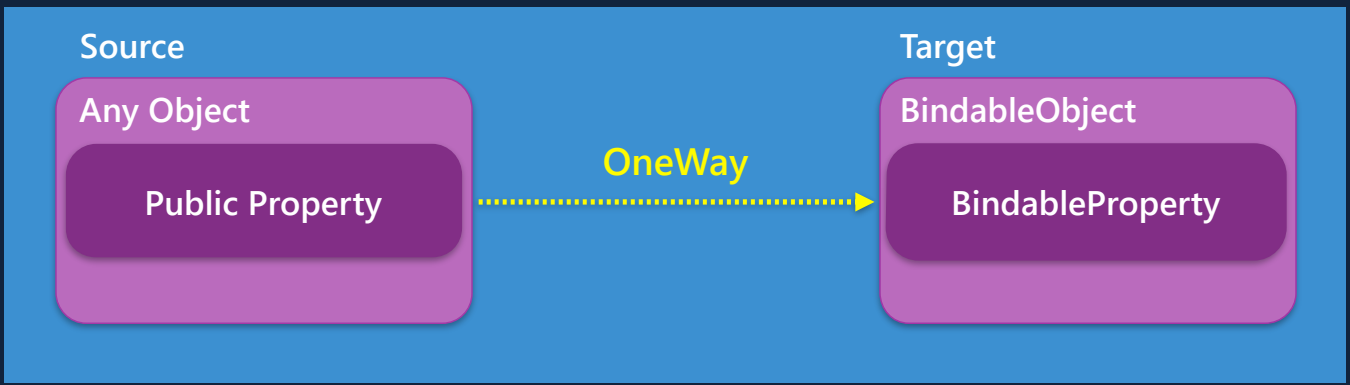
Creating two-way bindings

- Typically want data to flow in both directions
  - source > target (always happens)
  - target > source (optional)

Xamarin University

Binding Modes

Binding Mode controls the direction of the data transfer

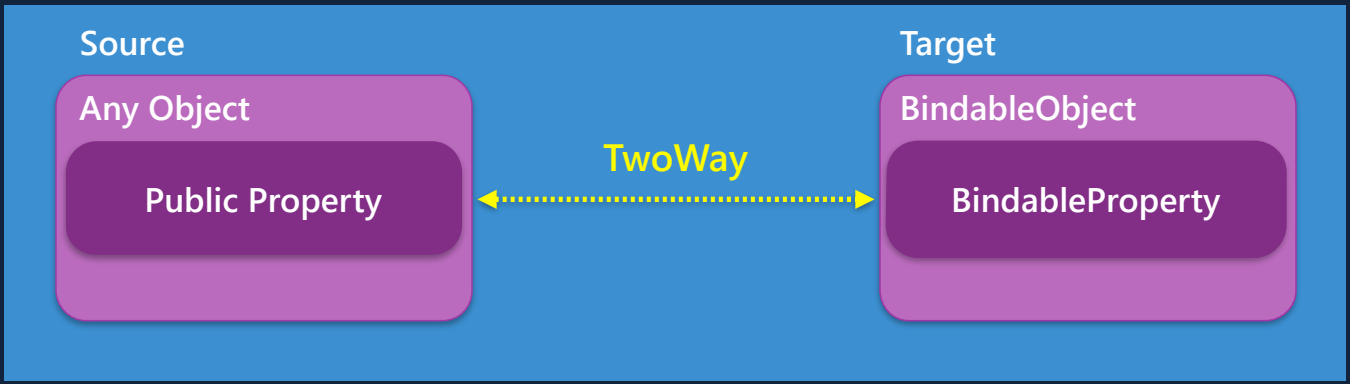


BindingMode.Default is the default value and it decides the mode based on the target property preference – either ***OneWay*** or ***TwoWay***

Xamarin University

Binding Modes

Binding Mode controls the direction of the data transfer

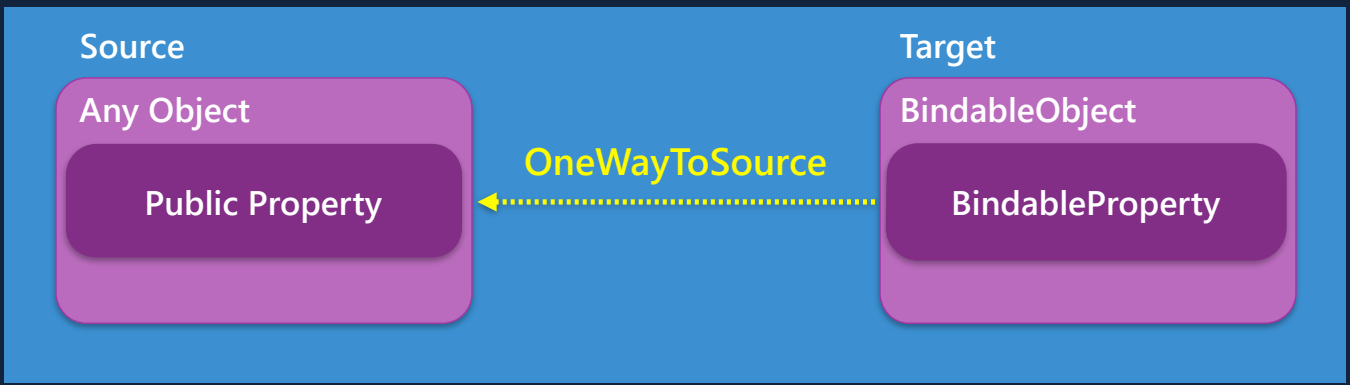


BindingMode.Default is the default value and it decides the mode based on the target property preference – either ***OneWay*** or ***TwoWay***

Xamarin University

Binding Modes

Binding Mode controls the direction of the data transfer



BindingMode.Default is the default value and it decides the mode based on the target property preference – either ***OneWay*** or ***TwoWay***

Xamarin University

Default Binding Mode

Default binding mode is property-specific, most are one-way by default with a few exceptions that default to two-way

```
DatePicker.Date
Entry.Text
ListView.SelectedItem
MultiPage<T>.SelectedIte
Picker.SelectedIndex
SearchBar.Text
Stepper.Value
Switch.IsToggled
TimePicker.Time
```

Xamarin University

Demo

Xamarin University

Property Changes

Xamarin University

Pushing changes to the UI

One-Way and Two-Way bindings always update the UI when the source property is changed

```
public class ComicBook {
      public ComicBook () {}

      public int SeriesNumber  { get; set; }
      public string SeriesName  { get; set; }
      public string Writer  { get; set; }
      public string Penciller  { get; set; }
      public string Inker  { get; set; }
      public string Colorist  { get; set; }
      public string Letterer  { get; set; }
      public string Editor  { get; set; }
      public string StoryTitle  { get; set; }
      public string Character { get; set; }
      public string ISBN { get; set; }
      public string Barcode { get; set; }
      public string ImageSmall { get; set; }
      public string Url { get; set; }
}
```

Update: comicBook.Writer = "Chris van Wyk";

Xamarin University

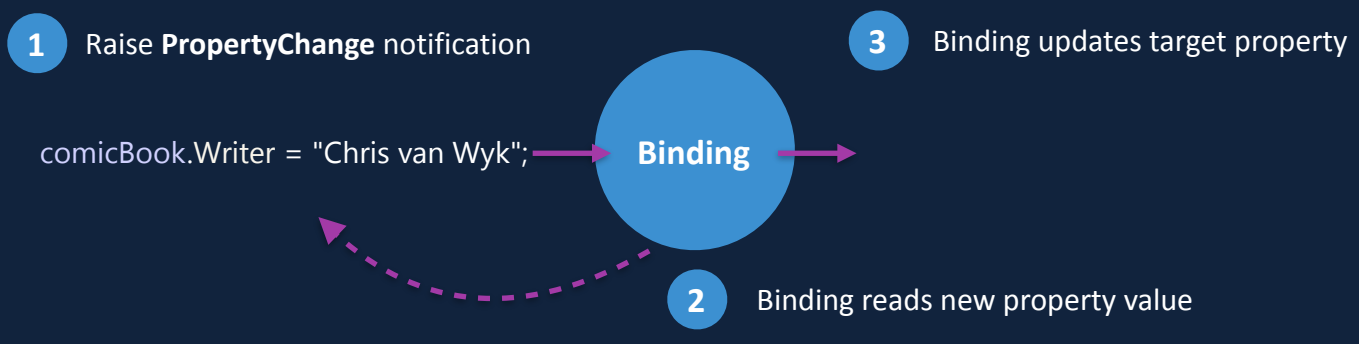
INotifyPropertyChanged

INotifyPropertyChanged provides change notification contract, should be implemented by any modifiable model object you bind to

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


INPC + Bindings

Binding will subscribe to the **PropertyChanged** event and update the target property when it sees the source property notification



Demo

Xamarin University

Summary

- Mapping Data to Visuals
- Creating Bindings in Code
- Creating Bindings in XAML
- Working with Binding Context
- Binding Modes
- Property Change Notifications

Xamarin University