

# Overview of Data Binding in Xamarin.Forms

---



**Gill Cleeren**

CTO XPIRIT BELGIUM

@gillcleeren [www.snowball.be](http://www.snowball.be)



# Overview



**A world without data binding**

**Data binding one-on-one**

**Binding modes and change notifications**



# A World without Data Binding

---



# How We Work Normally

XAML

Code-behind



# Displaying Data

```
private void BindData(Pie pie)
{
    PieNameLabel.Text = pie.PieName;
    PieImage.Source = pie.ImageUrl;
    DescriptionLabel.Text = pie.Description;
    PriceLabel.Text = pie.Price.ToString("c");
    InStockLabel.Text =
        pie.InStock ? "In stock" : "Not in stock";
}
```



```
PieNameEntry.TextChanged += (sender, e) =>  
    pie.pieName = PieNameEntry.Text;
```

## Updating Data Using Event Handlers



# Issues with This Approach



Error-prone



Mix of UI and non-UI code



Harder to maintain and test



# Data Binding One-on-one

---





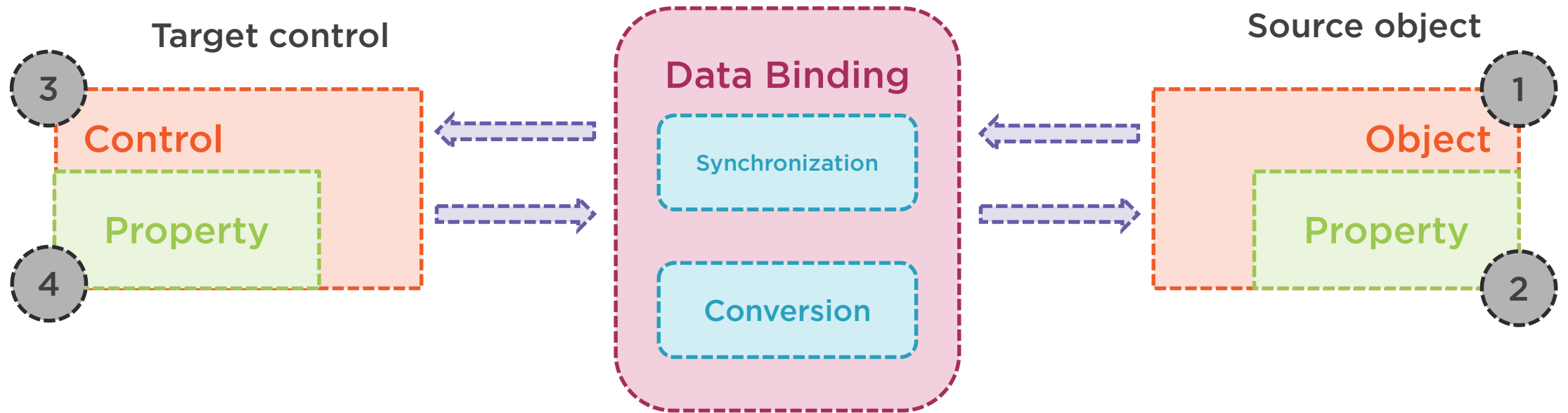
Data binding is the infrastructure that allows us to link the UI with data objects



Changes to the data that we bind to, can cause the UI to update automatically



# Data Binding



# Building Blocks

Source object

Source property

Target object

Target property



# Creating a Data Binding in Code

1 `Pie pie = new Pie () {};` *//could be loaded from a service*

3 `Entry pieNameEntry = new Entry();`

`Binding pieNameBinding = new Binding();`

`pieNameBinding.Source = pie;`

2 `pieNameBinding.Path = "PieName";`

4 `pieNameEntry.SetBinding  
    (Entry.TextProperty, pieNameBinding);`



# XAML Binding Syntax


The diagram illustrates the XAML Binding Syntax with four numbered annotations in dashed circles: 1 points to the ellipsis '...', 2 points to the 'Source' property, 4 points to the 'Binding' keyword, and 3 points to the opening tag '<TargetControl'. The code snippet is: `<TargetControl TargetProperty="{Binding SourceProperty, Source = ..., bindingProperties}" />`

```
<TargetControl  
    TargetProperty="{Binding SourceProperty, Source = ...,  
                    bindingProperties}" />
```



# Creating a Data Binding in XAML

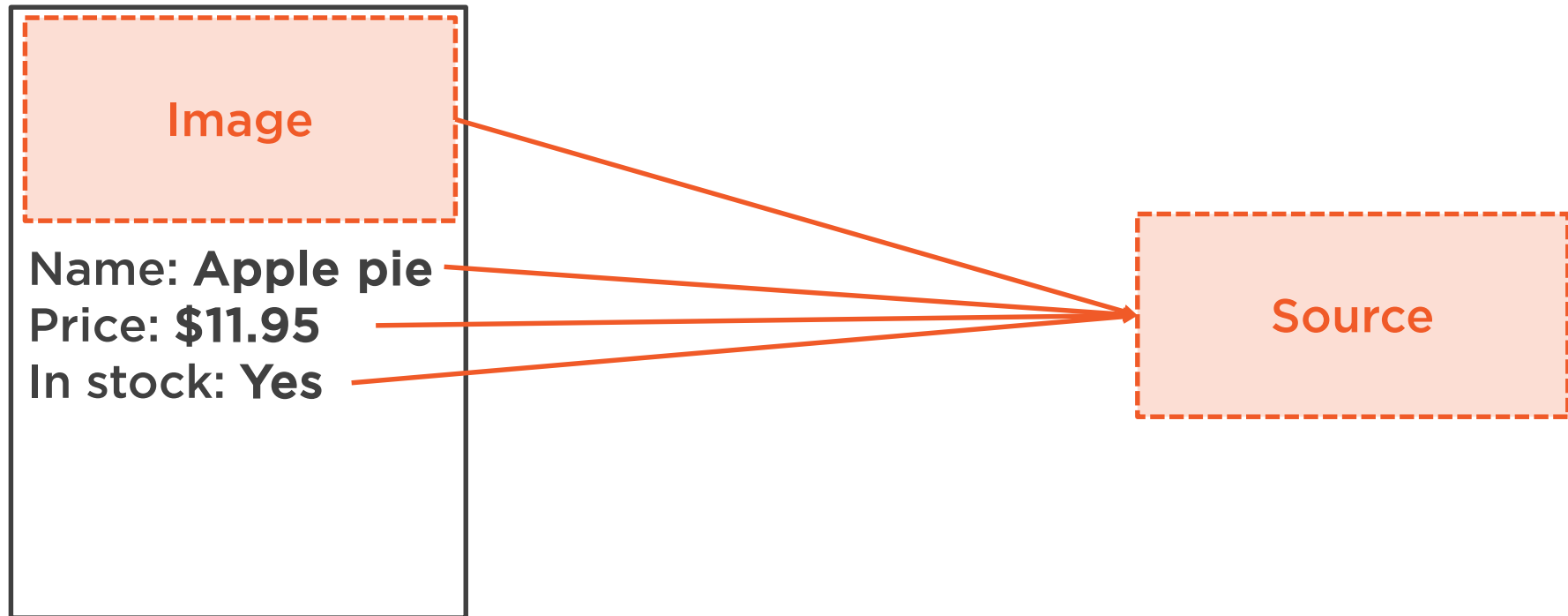
```
<Grid.Resources>  
    <t:Pie x:Key="applePie" />  
</Grid.Resources>  
...  
<Entry Text="{Binding Name,  
          Source={StaticResource applePie}}" />
```



A diagram consisting of an orange line that starts at the 'applePie' text within the 'Source={StaticResource applePie}' property of the <Entry> tag. The line extends horizontally to the right, then vertically upwards, and finally horizontally to the left, ending with an arrowhead pointing at the 'applePie' text within the 'x:Key="applePie"' property of the <t:Pie> tag inside the <Grid.Resources> block.

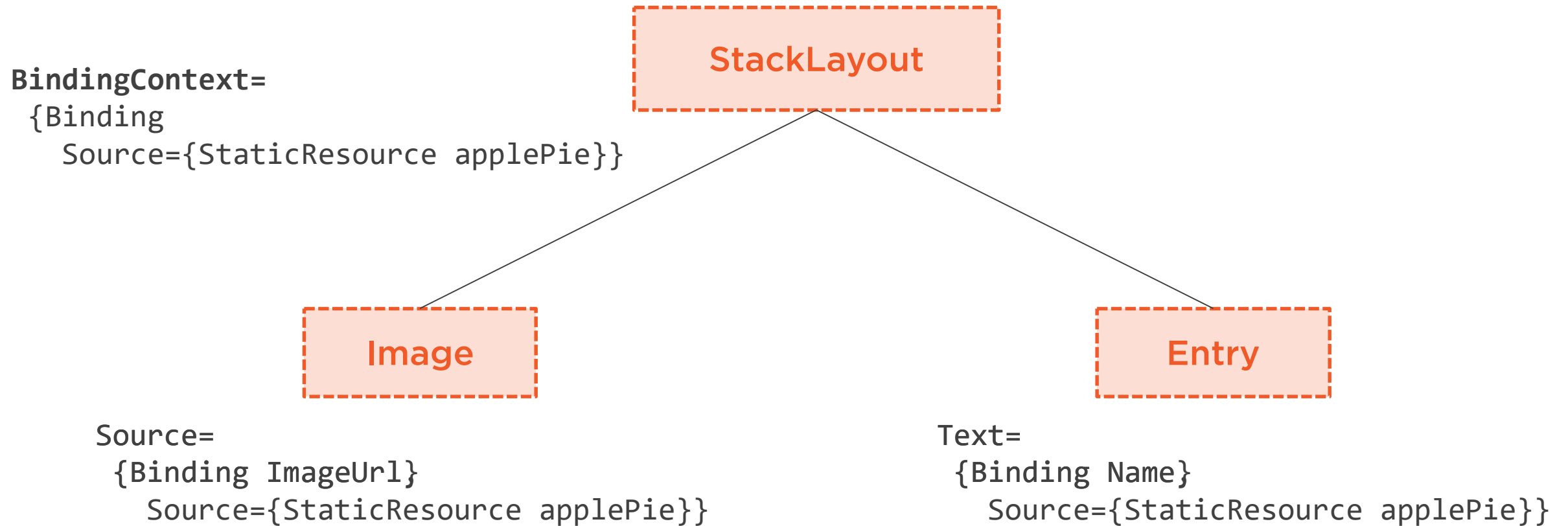


# The XAML Hierarchy

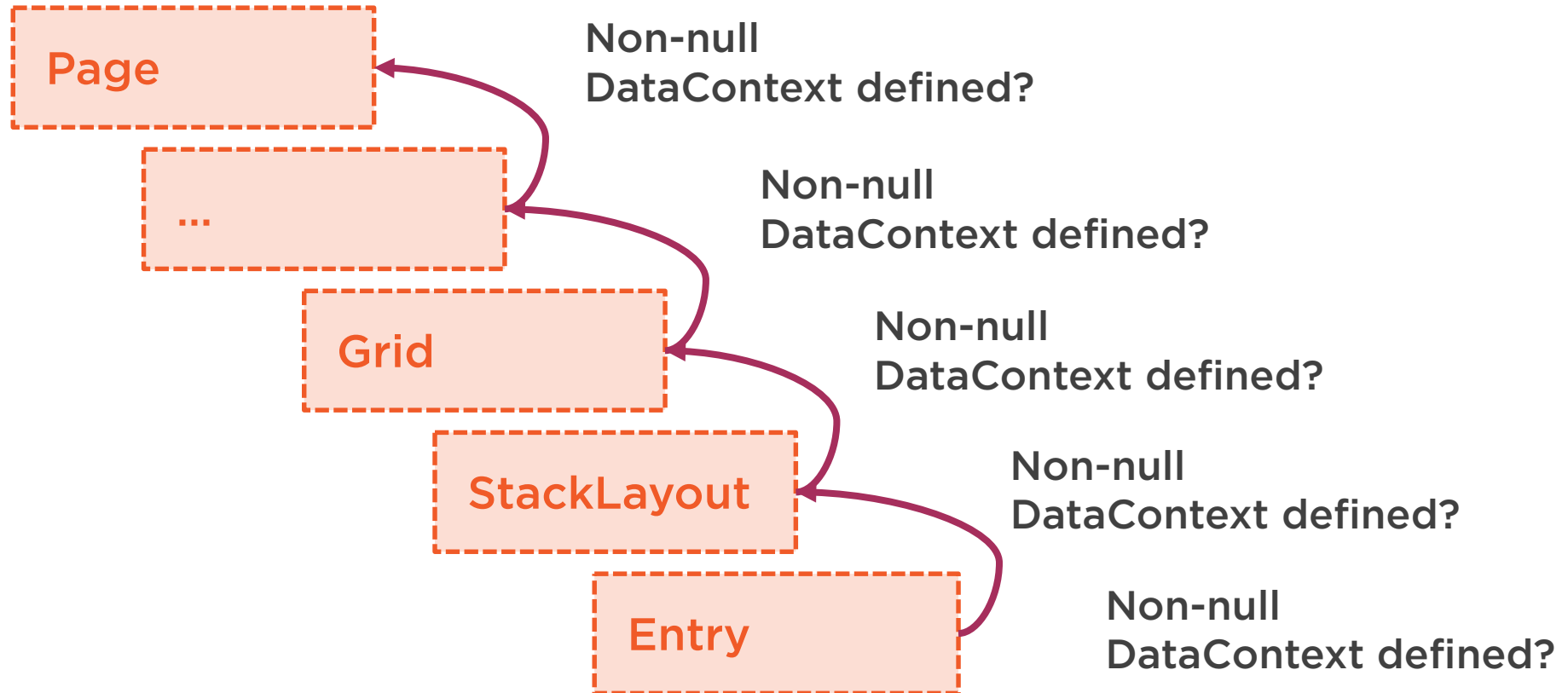




# Binding Multiple Elements



# XAML Tree Walking



```
var applePie = new Pie
{
    Id = 1,
    InStock = true,
    PieName = "Apple pie",
    Price = 20.95
};

MainStackLayout.BindingContext = applePie;
```

## Specifying the BindingContext



# Demo



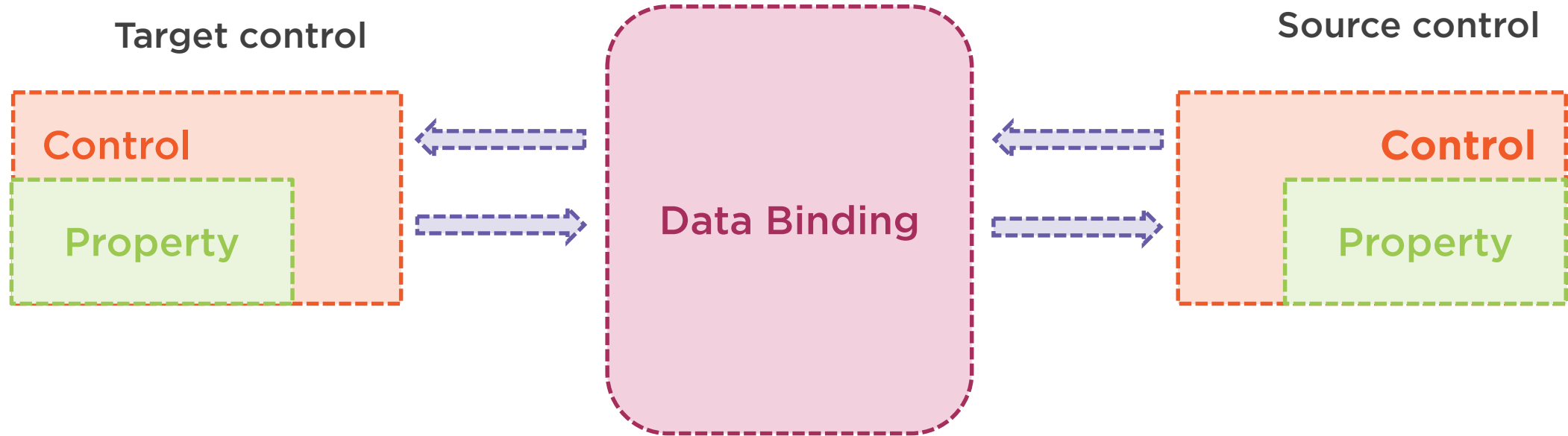
Creating a data binding in code

Creating a data binding in XAML

Working with the BindingContext



# View-to-view Bindings



# View-to-view Bindings

```
<Label Text="Pie size"  
      BindingContext="{x:Reference Name=slider}"  
      Text="{Binding Path=Value}" />
```

```
<Slider x:Name="slider"  
        Maximum="20" />
```



# Demo



## View-to-view bindings



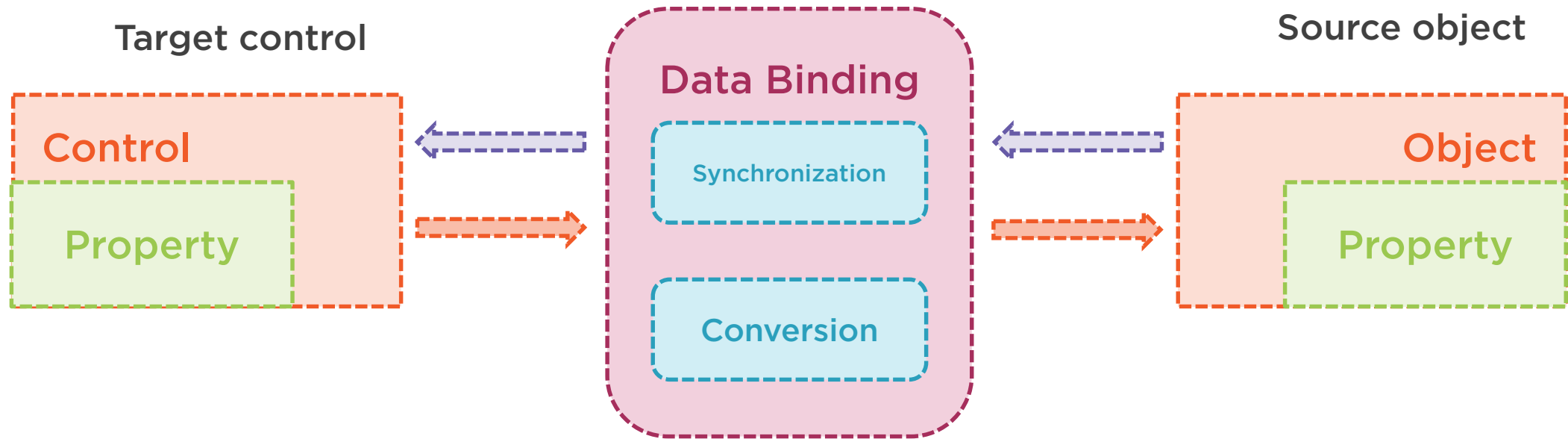
# Binding Modes and Change Notifications

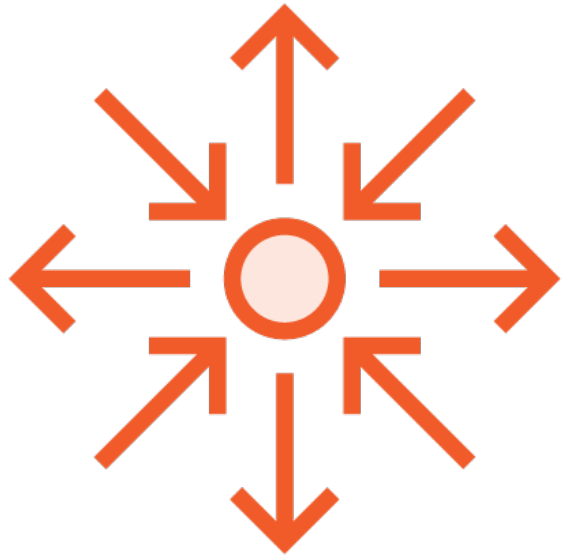
---





# Remember these Arrows?

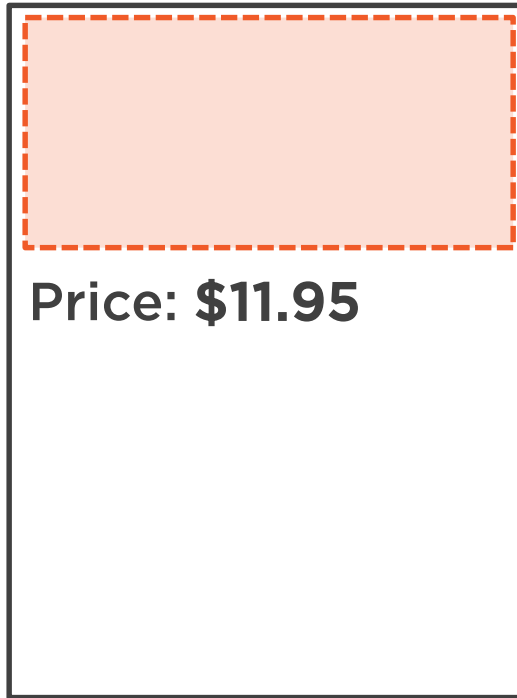




## Binding modes

- Default
- OneTime
- OneWay
- TwoWay
- OneWayToSource

# OneTime Binding

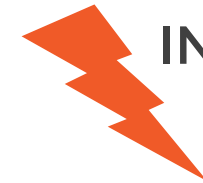


Text="{Binding Path=Price,  
Mode=OneTime}"



# Change Notifications



 **INotifyPropertyChanged**



```
public interface INotifyPropertyChanged
{
    event PropertyChangedEventHandler PropertyChanged;
}
```

---

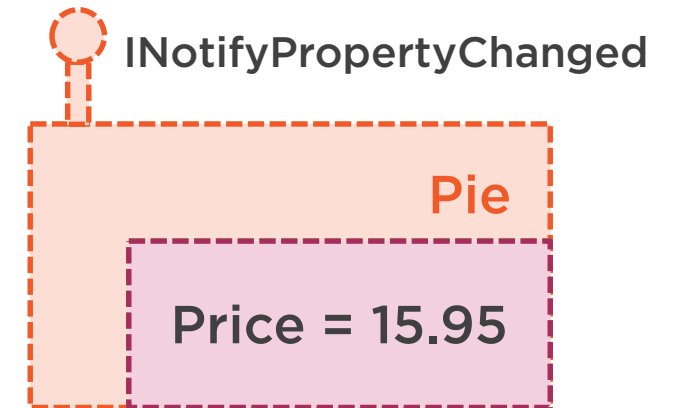
INotifyPropertyChanged



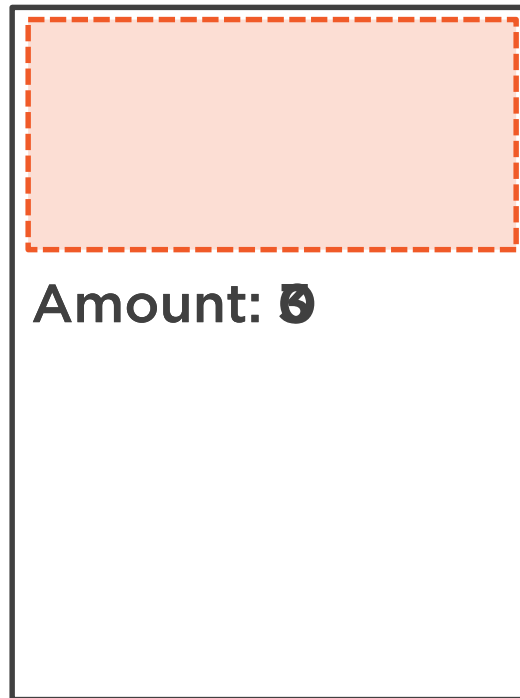
# OneWay Binding



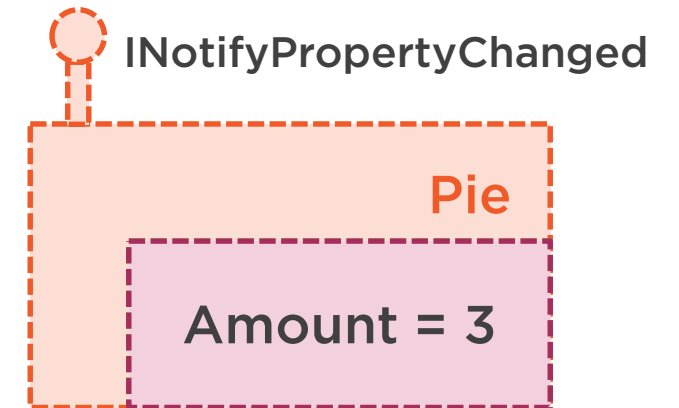
Text="{Binding Path=Price,  
Mode=**OneWay**}"



# TwoWay Binding



Text="{Binding Path=Amount,  
Mode=TwoWay}"





The mode depends on the property

Mostly one-way

Some exceptions default to TwoWay

- Entry.Text
- DatePicker.Text
- Stepper.Value
- ListView.SelectedItem
- ...



# Demo



## Implementing INotifyPropertyChanged

### Binding modes



# Summary



Creating data-driven screens is error-prone without data binding

Data binding is used mostly from XAML

Through binding modes, easy to register for changes





**Up next:**  
Creating a data-bound screen

