Binding to a Collection

Gill Cleeren
CTO XPIRIT BELGIUM

@gillcleeren www.snowball.be

Overview

Binding the ListView to a collection

Templating the ListView

Binding the ListView to a Collection

The ListView

Display list of data

Works with data binding

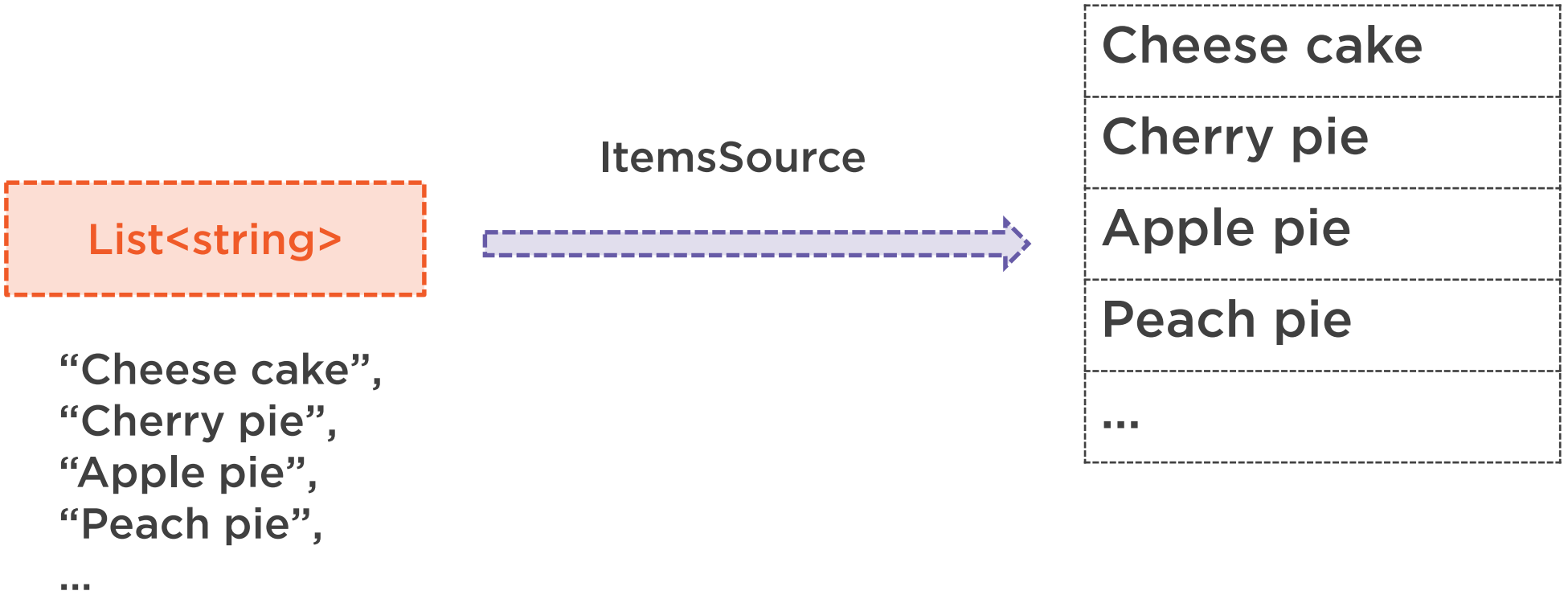
Cells

Same type per row

```
<ListView
    VerticalOptions="FillAndExpand“
    x:Name="PiesListView"
    CachingStrategy="RecycleElement" >
</ListView>
```

The ListView Control

The ItemsSource



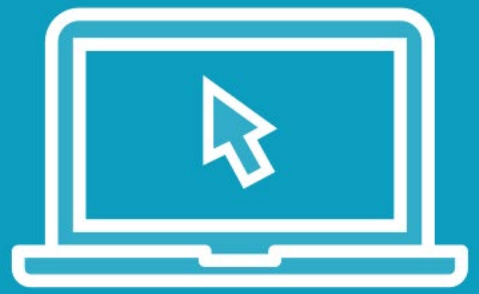
Binding via XAML

```
<ListView x:Name="PiesListView">
    <ListView.ItemsSource>
        <x:Array Type="{x:Type x:String}">
            <x:String>Apple Pie</x:String>
            <x:String>Strawberry Cheese Cake</x:String>
            <x:String>Strawberry Pie</x:String>
    </x:Array>
</ListView.ItemsSource>
```

```
PiesListView.ItemsSource =
    new string[]
      { "Apple Pie",
        "Strawberry Cheese Cake",
        "Strawberry Pie"
      };
```

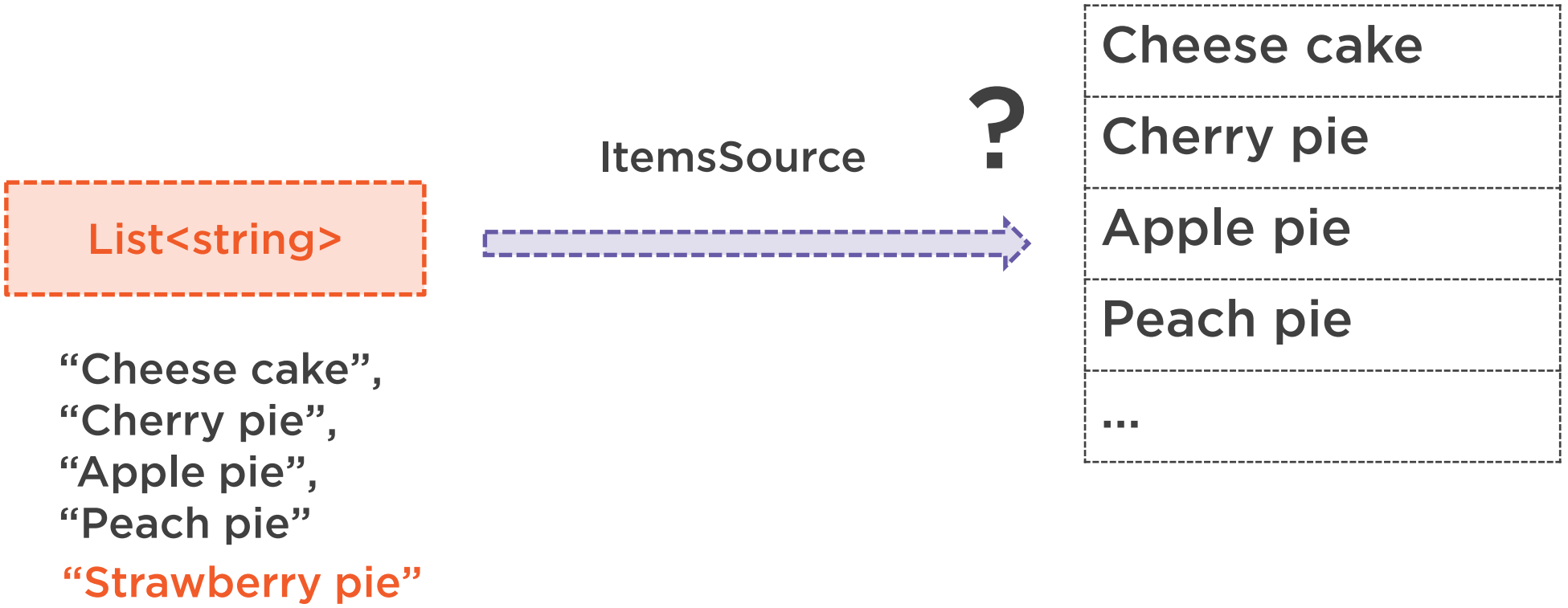
Setting ItemsSource in Code

Demo

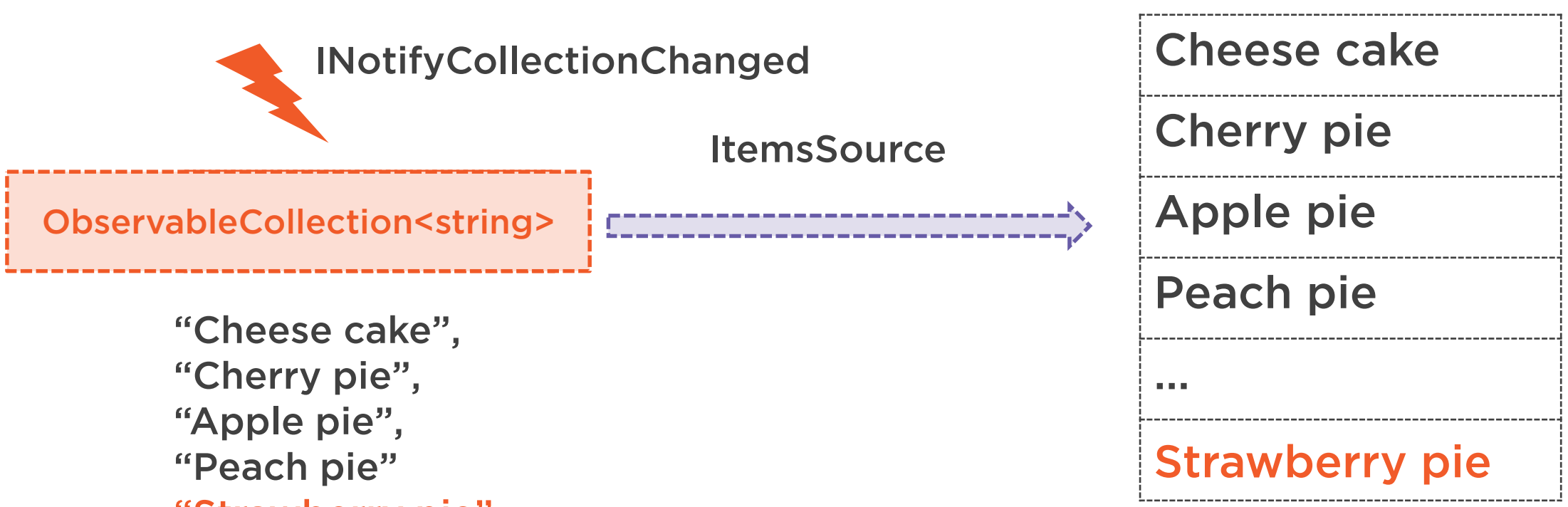


Binding to a collection

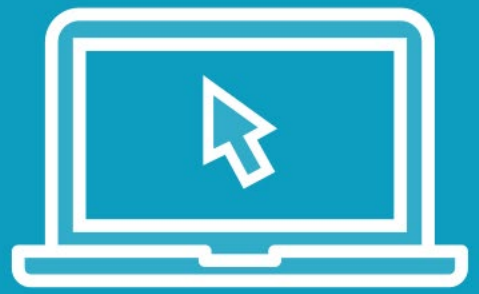
The ItemsSource



The ObservableCollection



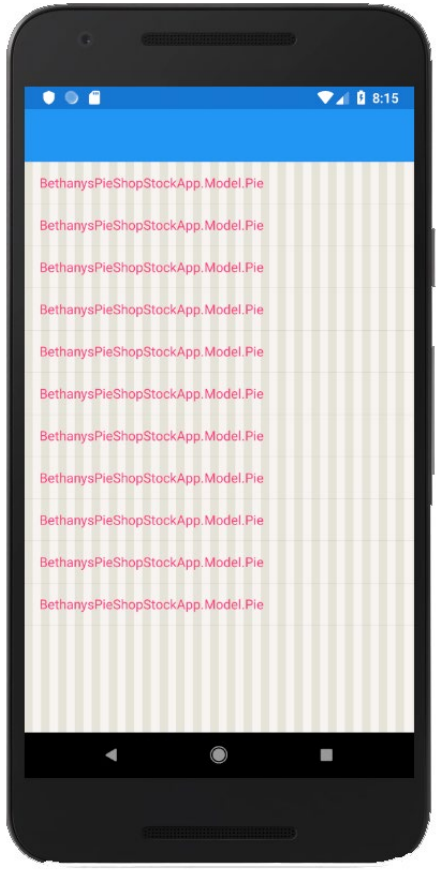
Demo



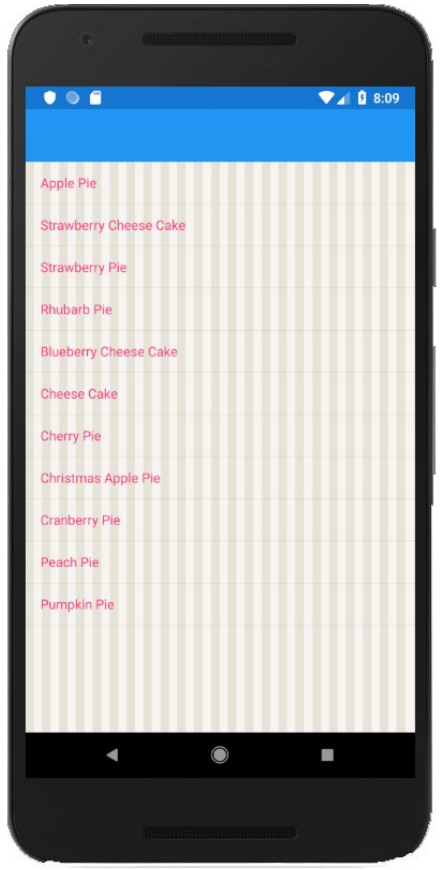
Working with ObservableCollection<T>

Templating the ListView

Default Behavior

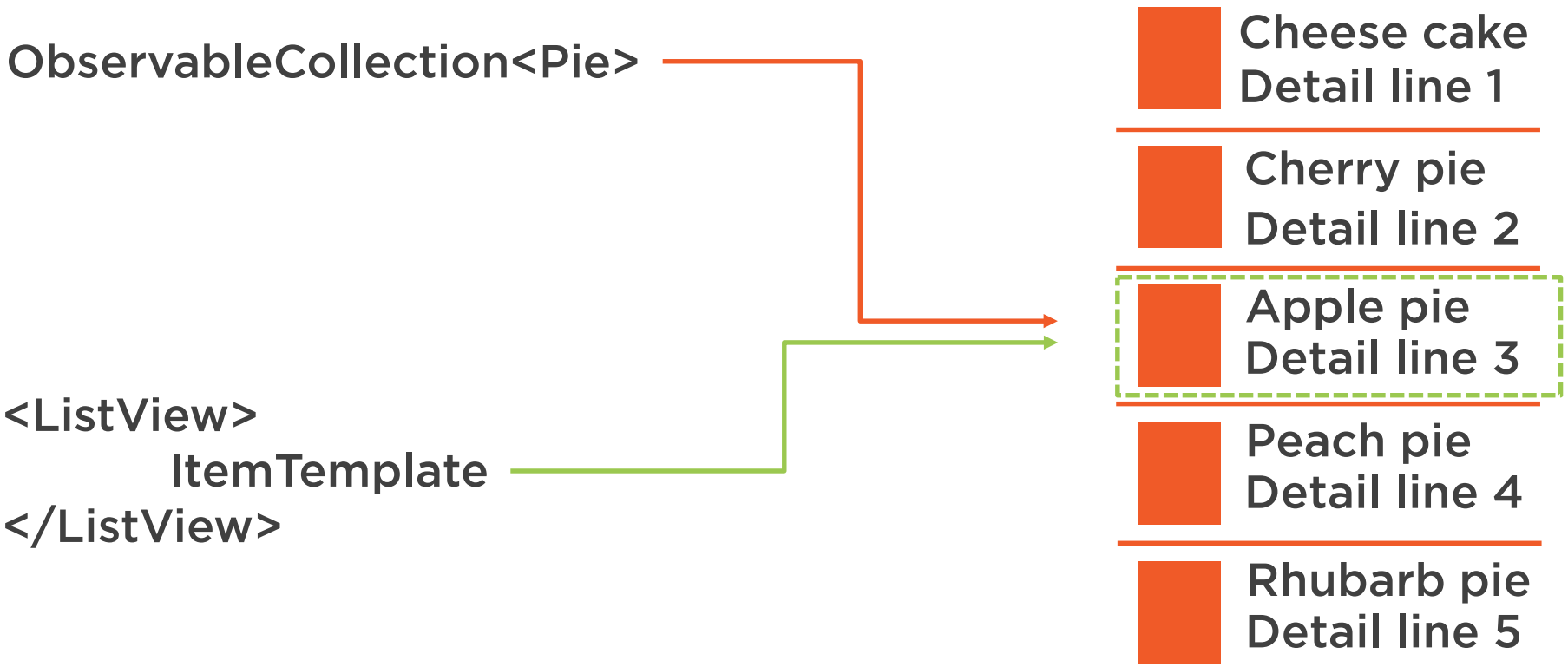


List<string>



List<Pie>

Templating


ObservableCollection<Pie>
<ListView>
ItemTemplate
</ListView>
Cheese cake
Detail line 1
Cherry pie
Detail line 2
Apple pie
Detail line 3
Peach pie
Detail line 4
Rhubarb pie
Detail line 5

Cell Templates

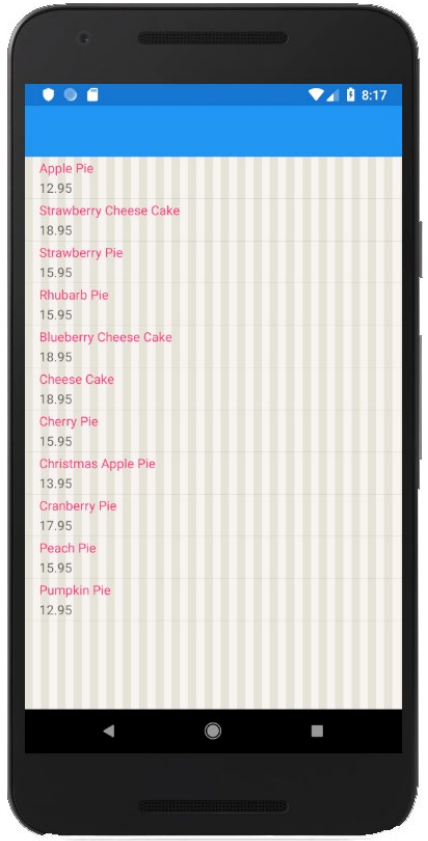
Built-in cell templates

TextCell

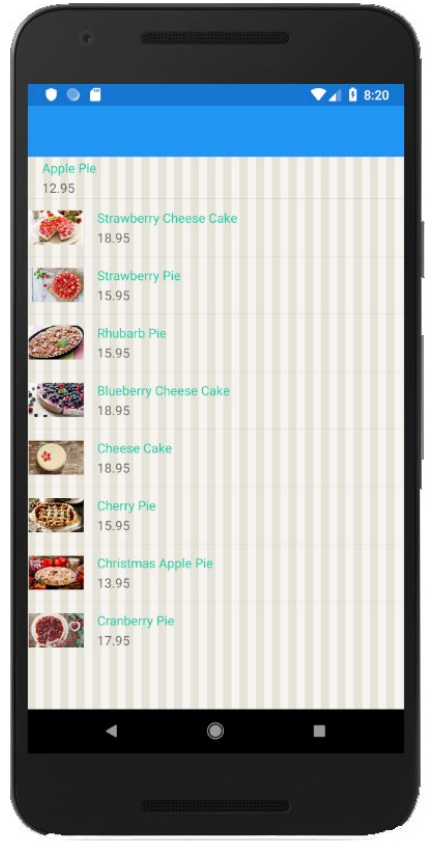
ImageCell

Custom cell

Using the Default Cell Templates



TextCell



ImageCell

Using an ItemTemplate

```
<ListView>
    <ListView.ItemTemplate>
        <DataTemplate>
            <TextCell></TextCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

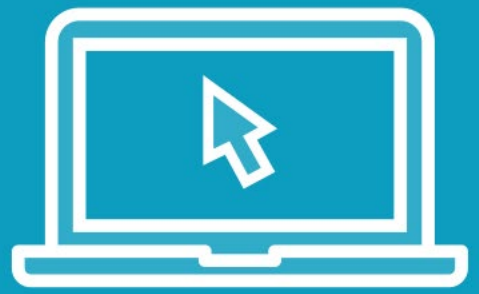
Data Binding within the Cell Template

```
<ListView.ItemTemplate>

    <DataTemplate>

        <TextCell
            Text="{Binding PieName}"
            Detail="{Binding Price}">

        </TextCell>

    </DataTemplate>

</ListView.ItemTemplate>
```

```
<ListView VerticalOptions="FillAndExpand"
    SelectedItem="{Binding SelectedPie, Mode=TwoWay}">
```

Binding the SelectedItem

Demo



Bind to objects

Templating

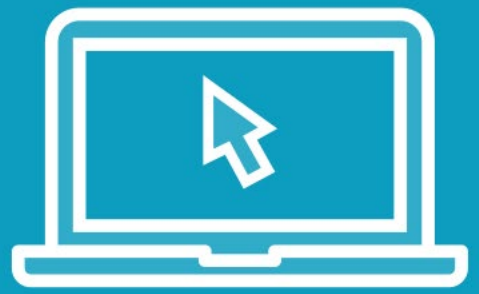
- TextCell
- ImageCell

Setting the SelectedItem via Binding

Creating a Custom ItemTemplate

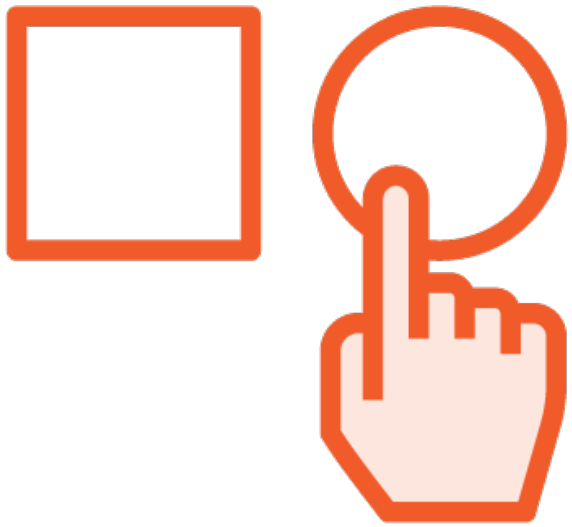
```
<ListView>
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <StackLayout>
                    <Label Text="{Binding PieName}" Margin="3" ></Label>
                </StackLayout>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Demo



Working with a custom cell template

Reusing the template across views



Data not always the same

DataTemplateSelector can be used

- Select template based on value

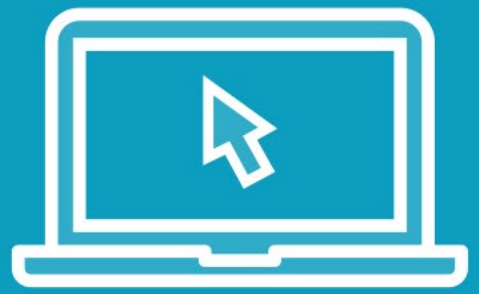
Creating a DataTemplateSelector

```
public class PieTemplateSelector : DataTemplateSelector
{
    protected override DataTemplate OnSelectTemplate
       (object item, BindableObject container)
    {
        return ((Pie)item).InPromotion ?
          PromoPieTemplate : RegularPieTemplate;
    }
}
```

```
<ListView
    ItemTemplate="{StaticResource localPieTemplateSelector}“
>
</ListView>
```

Applying the DataTemplateSelector

Demo



Creating and using a DataTemplateSelector

Summary

ListView is perfect fit for data binding

Cell templates allow for extensibility

Up next:
Advanced topics around data binding