Kelas Belajar Membangun Aplikasi Multi-platform (https://www.dicoding.com/academies/20)

Topik Modul 3: Menampilkan Data Pada ListView

Pendahuluan

Modul ini akan membahas bagaimana menggunakan kontrol ListView pada Xamarin Forms. ListView adalah salah satu kontrol yang banyak digunakan untuk menampilkan data pada layar ponsel yang memiliki keterbatasan ukuran. Untuk menampilkan data dari sumber data kedalam view/kontrol maka digunakan mekanisme data binding.

Binding Data yang bertipe List Of String

Mekanisme binding umum digunakan pada Xamarin Form. Binding adalah mengaitkan data yang ada pada data source (sumber data) ke kontrol tertentu. Ada dua macam jenis binding yaitu one-way binding dan two-way binding. One-way binding digunakan hanya untuk menampilkan data saja, sedangkan two-way binding digunakan untuk menampilkan dan mengedit data.

Practice #3.1 Menampilkan Data bertipe List Of String

1. Buat Xamarin **Cross Platform Portable solution** dengan nama **Modul3**. Kemudian pada project Portable tambahkan **halaman xaml baru (Form Xaml Page)** dengan nama **BindingListString.xaml**, kemudian tambahkan kode berikut:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Modul3.BindingListString">
  <ListView x:Name="listView" />
</ContentPage>
```

2. Pada file **BindingListString.xaml.cs** tambahkan kode C# berikut untuk menampilkan data berupa objek List kedalam kontrol ListView.

```
public BindingListString()
{
InitializeComponent();
List<string> items = new List<string> { "First", "Second", "Third" };
       listView.ItemsSource = items;
}
```

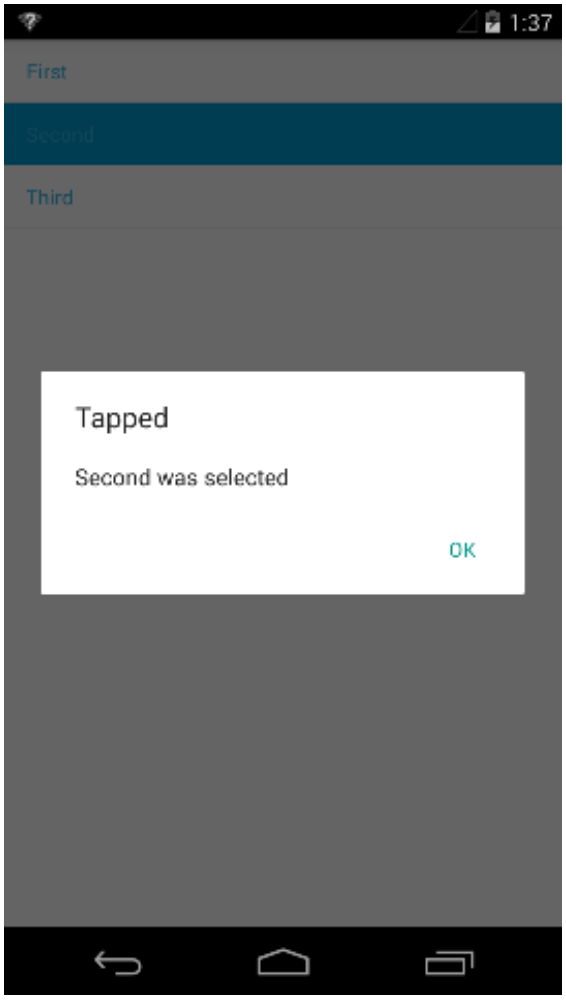
3. Anda juga dapat mengambil informasi data/item yang dipilih pada ListView, caranya adalah dengan menambahkan **event handler 'ItemTapped'** kedalam ListView.

```
        public BindingListString()
        {
            InitializeComponent();
            List<string> items = new List<string> { "First", "Second", "Third" };
            listView.ItemsSource = items;

            //untuk mengambil nilai item ketika diklik pada baris
            listView.ItemTapped += async (sender, e) =>
            {
                 await DisplayAlert("Tapped", e.Item.ToString() + " was selected", "OK");
                 ((ListView)sender).SelectedItem = null;
            };
        }
```

4. Setelah itu masuk ke halaman **App.xaml.cs** dan ubah kode program berikut di bagian constructor seperti berikut :

```
public App()
        {
            InitializeComponent();
            MainPage = new BindingListString();
        }
```

5. Tekan **F5** untuk menjalankan program pada emulator, hasilnya dapat dilihat pada gambar dibawah ini:



Practice #3.2 Menampilkan Data dari Objek Data Model

Pada contoh berikut dijelaskan cara menampilkan data dari objek data model yang sudah disiapkan sebelumnya.

1. Pada project Portable, tambahkan **folder baru** dengan nama **Models**, kemudian tambahkan class dengan nama **ListItem.cs**.

```
public class ListItem
{
    public string Title { get; set; }
    public string Description { get; set; }
}
```

Untuk class yang digunakan untuk Binding, pastikan menggunakan Public

2. Kemudian tambahkan **halaman xaml baru (Form Xaml Page)** dengan nama **BindingToDataModel.xaml**, kemudian tambahkan kode xaml berikut:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Modul3.BindingToDataModel">
  <ListView ItemsSource="{Binding ListItems}" ItemTapped="ListViewItemTapped">
    <ListView.ItemTemplate>
      <DataTemplate>
     <TextCell Text="{Binding Title}" Detail="{Binding Description}" DetailColor="Red" />
      </DataTemplate>
    </ListView.ItemTemplate>
  </ListView>
</ContentPage>
```

Pada halaman xaml diatas digunakan **ItemTemplate** untuk mengatur data yang akan ditampilkan pada kontrol ListView. Keyword Binding digunakan untuk mengikatkan data yang diambil dari objek data model kedalam kontrol ListView.

Element **TextCell** digunakan untuk menampilkan dua informasi yaitu title dan description.

3. Pada file **BindingToDataModel.xaml.cs** tambahkan kode berikut ini.

```
    public partial class BindingToDataModel : ContentPage
    {
        public BindingToDataModel()
        {
            InitializeComponent();
            BindingContext = new ListViewDataModelViewModel();
        }

        public class ListViewDataModelViewModel : BindableObject
        {
            private List<ListItem> lstItems;
            public ListViewDataModelViewModel()
            {
                lstItems = new List<ListItem>
                {
                    new ListItem { Title="Mystic", Description="Mystic team blue badge" },
                    new ListItem {Title="Instinct",Description="Instinct team yellow badge" },
                    new ListItem {Title="Valor",Description="Valor team red badge" }
                };
            }
            public List<ListItem> ListItems
            {
                get { return lstItems; }
                set {
                    lstItems = value;
                    OnPropertyChanged("ListItems");
                }
            }
        }
    }
```

Jika ada merah pada **ListItem** tambahkan **using Modul3.Models;**

Property **BindingContext** digunakan untuk mengarahkan sumber data yang akan ditampilkan pada halaman xaml.

Class **ListViewDataModelViewModel.cs**. akan digunakan untuk membuat objek ListItems yang akan ditampilkan kedalam kontrol ListView.

Pada kode diatas dapat dilihat bahwa class **ListViewDataModelViewModel** diturunkan dari class BindableObject, ini bertujuan agar kita dapat menggunakan method **OnPropertyChanged()** yang akan memberi tahu program bahwa ada objek yang berubah.

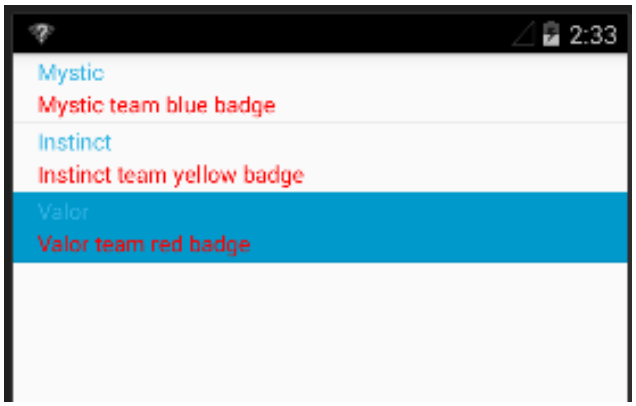
4. Untuk mengambil informasi dari data yang dipilih pada kontrol ListView, anda dapat menambahkan **event ItemTapped** ke dalam class **BindingToDataModel** seperti ditunjukan pada kode berikut:

```
        private async void ListViewItemTapped(object sender, ItemTappedEventArgs e)
        {
            ListItem item = (ListItem)e.Item;
            await DisplayAlert("Tapped", item.Title.ToString() + " was selected", "OK");
            ((ListView)sender).SelectedItem = null;
        }
```

5. Setelah itu masuk ke halaman **App.xaml.cs** dan ubah kode program berikut di bagian constructor seperti berikut :

```
public App()
        {
            InitializeComponent();
            MainPage = new BindingToDataModel();
        }
```

6. Tekan tombol **F5** untuk menjalankan aplikasi pada android emulator.



Practice #3.3 Menampilkan Gambar pada Cell

Pada contoh yang sebelumnya anda sudah menggunakan ListView ItemTemplate untuk menampilkan informasi berupa Text dan Detail. Kontrol ListView pada Xamarin juga menyediakan template yang menampilkan tidak hanya data berupa text, pada contoh dibawah ini kita akan mencoba untuk menampilkan data berupa gambar pada kontrol ListView.

1. Pada project portable, tambahkan halaman xaml baru dengan nama **ListViewImageCell.xaml**.

2. Kemudian tambahkan kode xaml berikut ini:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Modul3.ListViewImageCell">
  <ListView x:Name="listView" ItemsSource="{Binding ListItems}">
    <ListView.ItemTemplate>
      <DataTemplate>
        <ImageCell ImageSource="{Binding Source}" Text="{Binding Title}" Detail="{Binding Descrip
tion}" />
      </DataTemplate>
    </ListView.ItemTemplate>
  </ListView>
</ContentPage>
```

Pada kode xaml diatas dapat dilihat bahwa elemen Data Template yang digunakan berbeda dengan contoh sebelumnya yang menggunakan element **TextCell**, pada kasus ini digunakan element **ImageCell** untuk menampilkan data berupa image dan text.

3. Pada folder Models **modifikasi class ListItem.cs** yang sebelumnya sudah dibuat. Tambahkan property Source yang akan digunakan untuk menyimpan informasi gambar.

```
    public class ListItem
    {
        public string Source { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
    }
```

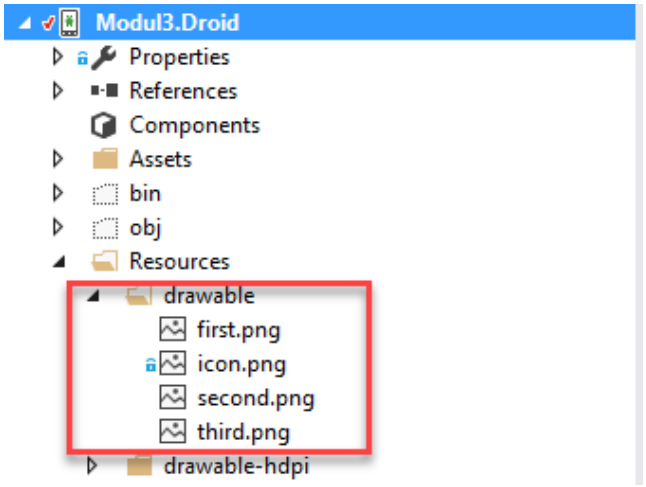
4. Pada file **ListViewImageCell.xaml.cs** tambahkan kode berikut ini

```
    public partial class ListViewImageCell : ContentPage
    {
        public ListViewImageCell()
        {
            InitializeComponent();
            BindingContext = new ListViewImageCellViewModel();
        }


        public class ListViewImageCellViewModel : BindableObject
        {
            private List<ListItem> listItems;
            public List<ListItem> ListItems
            {
                get { return listItems; }
                set
                {
                    listItems = value;
                    OnPropertyChanged("ListItems");
                }
            }

            public ListViewImageCellViewModel()
            {
                listItems = new List<ListItem>
                {
                    new ListItem { Source="first.png", Title="Mystic", Description="Mystic team b
lue badge" },
                    new ListItem { Source="second.png", Title="Instinct",Description="Instinct te
am yellow badge" },
                    new ListItem { Source="third.png", Title="Valor",Description="Valor team red
 badge" }
                };
            }
        }
    }
```

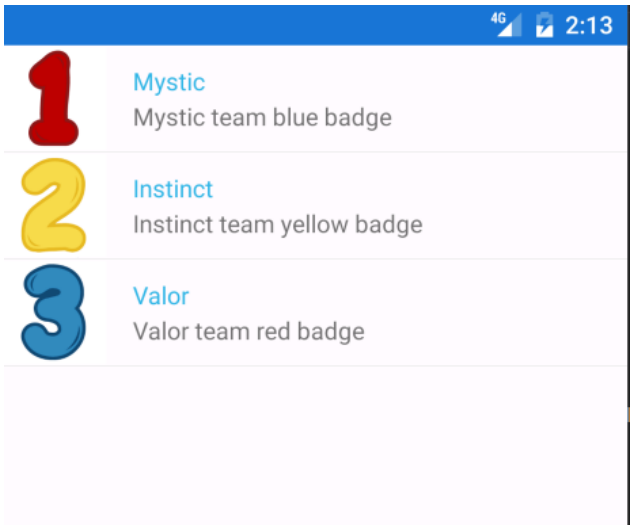
5. Tambahkan juga class dengan nama **ListViewImageCellViewModel.cs**. Class ini berisi objek-objek yang akan ditampilkan pada kontrol **ListView**.

6. Kode diatas mirip dengan contoh sebelumnya, hanya ada tambahan inisialisasi satu property baru yaitu Source yang berisi nama file image yang akan ditampilkan.

7. Untuk menambahkan image yang akan ditampilkan, tambahkan image yang akan ditampilkan kedalam project Droid (untuk aplikasi android) di folder **Resources\drawable**. Pada contoh berikut ditambahkan tiga file bertipe .png kedalam folder tersebut. **Anda dapat download asset tersebut disini (https://gifdicoding.blob.core.windows.net/academyxamarin/123.zip)**



8. Kemudian tekan tombol F5 untuk menjalankan program pada emulator, hasil dari aplikasi yang sudah dibuat dapat dilihat pada kode berikut ini. Jangan lupa untuk mengganti inisalisasi page yang diload oleh aplikasi pada **App.xaml.cs**



Practice #3.4 Kustomisasi Baris pada ListView

Selain menampilkan teks dan gambar pada kontrol ListView yang sudah dibuat pada contoh sebelumnya, kita juga dapat membuat tampilan sendiri sesuai dengan kebutuhan. Pada contoh dibawah ini ditunjukan bagaimana cara menampilkan ListView yang mempunyai baris yang sudah dikustomisasi.

1. Pada project portable, tambahkan halaman xaml baru dengan nama **ListViewCustom.xaml**. Kemudian tambahkan xaml berikut ini.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Modul3.ListViewCustom">
  <ListView x:Name="listView" ItemsSource="{Binding ListItems}" RowHeight="80" BackgroundColor="B
lack">
    <ListView.ItemTemplate>
      <DataTemplate>
        <ViewCell>
          <StackLayout HorizontalOptions="StartAndExpand" Orientation="Horizontal" Padding="25,1
0,55,15">
            <StackLayout HorizontalOptions="StartAndExpand" Orientation="Vertical">
              <Label HorizontalOptions="Start" FontSize="20" FontAttributes="Bold" TextColor="Whi
te" Text="{Binding Title}"/>
              <Label HorizontalOptions="Start" FontSize="12" FontAttributes="Bold" TextColor="Whi
te" Text="{Binding Description}"/>
            </StackLayout>
            <Label HorizontalOptions="End" FontSize="25" TextColor="Aqua" Text="{Binding Price}"
 />
          </StackLayout>
        </ViewCell>
      </DataTemplate>
    </ListView.ItemTemplate>
  </ListView>
</ContentPage>
```

Pada kode xaml diatas dapat dilihat bahwa kontrol ListView dapat mempunyai ItemTemplate yang berupa ViewCell. Element ViewCell ini sangat fleksible sehingga dapat diisi dengan element lain. Pada contoh diatas dapat dilihat bahwa ViewCell diisi dengan tiga kontrol label yang disusun menggunakan StackLayout.

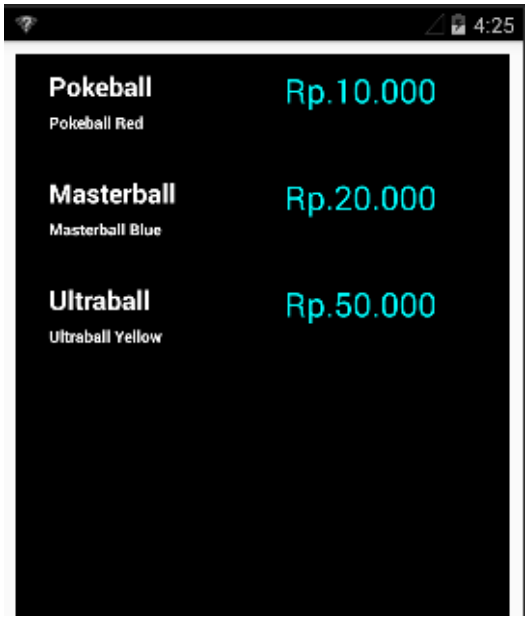
2. Pada folder Models **modifikasi class ListItem** yang sebelumnya sudah dibuat dengan menambahkan property baru yaitu **Price**.

```
    public class ListItem
    {
        public string Source { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public string Price { get; set; }
    }
```

3. Tambahkan kode pada file **ListViewCustom.xaml.cs** sebagai berikut:

```
    public partial class ListViewCustom : ContentPage
    {
        public ListViewCustom()
        {
            InitializeComponent();
            BindingContext = new ListViewCustomViewModel();
        }

        public class ListViewCustomViewModel : BindableObject
        {
            private List<ListItem> listItems;

            public List<ListItem> ListItems
            {
                get { return listItems; }
                set
                {
                    listItems = value;
                    OnPropertyChanged("ListItems");
                }
            }

            public ListViewCustomViewModel()
            {
                listItems = new List<ListItem>
                {
                    new ListItem { Title="Pokeball", Description="Pokeball Red", Price="Rp.10.00
0" },
                    new ListItem {Title="Masterball",Description="Masterball Blue",Price="Rp.20.0
00" },
                    new ListItem {Title="Ultraball",Description="Ultraball Yellow",Price="Rp.50.0
00" }
                };
            }
        }
```

4. Untuk menjalankan program tekan tombol **F5**, maka anda akan dapat melihat hasilnya pada android emulator. Jangan lupa untuk mengganti inisalisasi page yang diload oleh aplikasi pada **App.xaml.cs**



Ke Halaman Kelas (https://www.dicoding.com/academies/20)

Selesai & Lanjutkan → (https://www.dicoding.com/academies/20/tutorials/698?from=671)

← Sebelumnya (https://www.dicoding.com/academies/20/tutorials/707)