Lab 6 – ListView

Mr. Yousif Garabet Arshak
Computer Science Department
University of Zakho
yousif.arshak@uoz.edu.krd

Outlines

- **ListView**
- **Use cases**
- **Components**
- **Functionality**

Xamarin.Forms ListView

ListView is a view for presenting lists of data, especially long lists that require scrolling.

CollectionView is a view for presenting lists of data using different layout specifications. It aims to provide a more flexible, and performant alternative to **ListView**. For more information, see **Xamarin.Forms CollectionView**.

ListView Use cases

- A ListView control can be used in any situation where you're displaying scrollable lists of data. The ListView class supports context actions and data binding.
- The ListView control shouldn't be confused with the TableView control. The TableView control is a better option whenever you have a non-bound list of options or data because it allows predefined options to be specified in XAML. For example, the iOS settings app, which has a mostly predefined set of options, is better suited to use a TableView than a ListView.
- The ListView class doesn't support defining list items in XAML, you must use the ItemsSource property or data binding with an ItemTemplate to define items in the list.
- A ListView is best suited for a collections consisting of a single data type. This requirement is because only one type of cell can be used for each row in the list. The TableView control can support multiple cell types, so it is a better option when you need to display multiple data types.

ListView Components

Headers and footers

Header and footer components display at the beginning and end of a list, separate from list's data. Headers and footers can be bound to a separate data source from the ListView's data source.

Groups

Data in a ListView can be grouped for easier navigation. Groups are typically data bound. The following screenshot shows a ListView with grouped data:

Cells

Data items in a ListView are called cells. Each cell corresponds to a row of data. There are built-in cells to choose from, or you can define your own custom cell. Both built-in and custom cells can be used/defined in XAML or code.

•Built-in cells, such as the TextCell and ImageCell, correspond to native controls and are especially performant.

- •A TextCell displays a string of text, optionally with detail text. Detail text is rendered as a second line in a smaller font with an accent color.
- •An ImageCell displays an image with text. Appears as a TextCell with an image on the left.

•Custom cells are used to present complex data. For example, a custom cell could be used to present a list of songs that includes the album and artist.

Functionality

The ListView class supports a number of interaction styles.

•Pull-to-refresh allows the user to pull the ListView down to refresh the contents.

•Context actions allow the developer to specify custom actions on individual list items. For example, you can implement swipe-to-action on iOS, or long-tap actions on Android.

•Selection allow the developer to attach functionality to selection and deselection events on list items.

- XAML →

```
<CarouselView x:Name="MyCarouselView" HeightRequest="400" IsScrollAnimated="True"
>
          <CarouselView.ItemTemplate>
            <DataTemplate>
              <Image Source="{Binding .}"/>
            </DataTemplate>
          </CarouselView.ItemTemplate>
        </CarouselView>
```

- C# →

```
var images = new List<string>
          {
            “image1.jpg",
            "image2.png",
            "image1.jpg"
          };
          MyCarouselView.ItemsSource = images;
```

Exercises

1- Create Fruit list in ListView.

2- Create Country list in ListView and show Country Name, Image, Capital

Any Questions?