**CHƯƠNG 6**

**Tạo một Scrollable**

Tạo một danh sách dài các mặt hàng là một trong những dịch vụ quan trọng của giao diện người dùng di động. Các sản phẩm bị thông tin hạn chế trên màn hình điện thoại di động làm cho hiển thị dữ liệu dạng lưới là một thách thức và dẫn

để một cách sử dụng rộng rãi và sáng tạo một danh sách. Nhóm các mục, cuộn, thao tách, và hình ảnh làm cho danh sách trở thành một trong những công cụ ràng buộc dữ liệu linh hoạt nhất và có thể sử dụng lại có sẵn.

Chương này bao gồm lớp ListView cho phép chúng ta tạo một cuộn

danh sách lựa chọn. Các cân nhắc chính bao gồm liên kết với một mảng hoặc mô hình dữ liệu, xử lý lựa chọn mục, tùy chỉnh giao diện của các hàng, nhóm tiêu đề và hiệu suất của chúng.

Xamarin.Forms List View

Danh sách trong Xamarin.Forms được tạo bằng điều khiển ListView được liên kết với một mảng hoặc kiểu dữ liệu. Lớp XViewarin.Forms ListView cung cấp một danh sách có thể cuộn, có thể lựa chọn. Khi liệt kê các hàng có thể tùy chỉnh bằng cách sử dụng bố cục, hình ảnh và chế độ xem như nút. ListView hỗ trợ nhóm : tiêu đề, chân trang, danh sách nhảy và kéo để làm mới. Xóa và áp dụng các hoạt động để liệt kê các hàng được hỗ trợ bằng cách sử dụng tác vụ bối cảnh. Các danh sách trong chương này là chỉ đọc, có nghĩa là chúng bị ràng buộc với một nguồn dữ liệu để xem và chọn không cho phép ghi, thay đổi nội dung. Chúng tôi chạm vào một số danh sách có thể chỉnh sửa Các kỹ thuật giao diện người dùng trong Danh sách tùy chỉnh của Rows , nhưng không bao gồm các thay đổi đối với mô hình dữ liệu hoặc liên kết dữ liệu hai chiều để các thay đổi này có thể được phản ánh trong danh sách. Để liên kết dữ liệu ListView có thể xem Chương [9](https://translate.googleusercontent.com/translate_f) .Chúng tôi có thể liên kết trực tiếp vào một bộ sưu tập hoặc danh sách bằng cách sử dụng thuộc tính ItemSource với mẫu danh sách mặc định. Chúng tôi cũng có thể liên kết với các mô hình dữ liệu và tạo các hàng tùy chỉnh với lớp adaptor built-in của ListView gọi là ItemTemplate. Hãy thử hai cách tiếp cận:

MụcSource và ItemTemplate. Bắt đầu với ItemSource, hãy để chúng tôi tiến hành bằng cách ràng buộc ListView đến một nguồn dữ liệu đơn giản.

Tạo một danh sách chuỗi ký tự

Việc thực hiện ListView đơn giản nhất được gắn vào một danh sách của Strings. Khởi tạo một lớp ListView trên trang của bạn và trỏ nó đến nguồn dữ liệu mặc định sử dụng thuộc tính ItemSource:

<ListView ItemSource = "{Binding.}" />

ItemSource được định nghĩa là BindingContext cục bộ, được biểu thị bằng một khoảng thời gian. Chúng ta sẽ sớm xác định BindingContext để chứa danh sách các mục trong mã phía sau. Điển hình là bạn sẽ muốn sử dụng XAML ItemTemplate và có thể là ViewModel để liên kết, nhưng chúng tôi đang bắt đầu đơn giản vớilưới và BindingContext của trang. ItemSource hoạt động (thêm về ràng buộc dữ liệu trong Chương [9](https://translate.googleusercontent.com/translate_f)).

Tiếp theo, trong hàm tạo của lớp chính, ví dụ như ListViewStrings trong liệt kê [6-1](https://translate.googleusercontent.com/translate_f#21), khai báo chuỗi danh sách lớp của bạn được gọi là các mục sau khi khởi tạo LaunchizeComponent ();. Mục dân cư và

đặt tất cả vào BindingContext của ContextPage.

***6-1.***Xây dựng ContentPage cho một ListView

public ListViewStrings()

{

InitializeComponent();

List Items = new List { "First",

"Second",

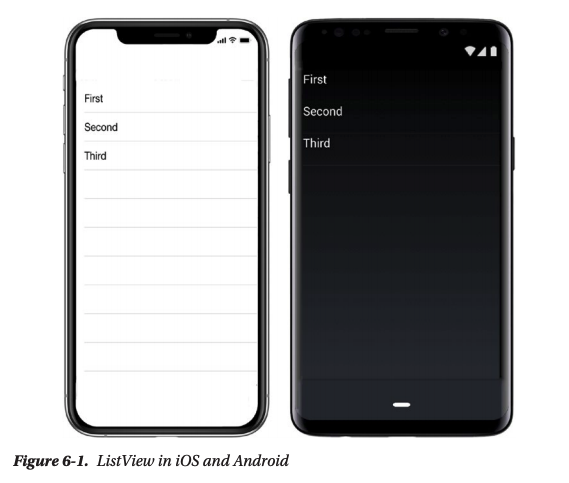
"Third"

};

BindingContext = Items;

}

Sử dụng bố cục ListView mặc định, mỗi mục trong danh sách sẽ là một ô duy nhất được sử dụng.TextCell hiển thị một dòng văn bản.



**Selecting an Item**

Có hai sự kiện để sử dụng trong lựa chọn item : ItemTapped và ItemSelected. Cả hai có thể xảy ra khi người dùng chạm một ô trong ListView. Sự khác biệt giữa chúng là khi một danh sách cho phép nhiều hơn là chỉ khai thác và các mục có thể được chọn và không được chọn. Trong các danh sách đơn giản, nơi không có sự ngăn cản của các hàng (như ví dụ ở đây), có rất ít sự khác biệt giữa chúng. ItemTapped là đơn giản nhất. Nó kích hoạt dưới dạng một sự kiện chuyển động khi người dùng nhấp vào một hàng danh sách:

<ListView ItemsSource = "{Binding Items}" ItemTapped = "ListViewItemTapped" />

Sự kiện ItemTapped phải được triển khai trong C # :

async void ListViewItemTapped (object sender, ItemTappedEventArgs e)

{

string mục = (string) e.Item;

await DisplayAlert ( "Tapped", item + "was selected", "OK");

}

Sự kiện ItemSelected phản ứng với sự thay đổi trạng thái của hàng chọn và

xảy ra khi một hàng được chọn hoặc không được chọn bởi người dùng:

<ListView ItemsSource = "{Binding Items}" ItemSelected = "ListViewItem

Selected "/>

Sự kiện ItemSelected phải được triển khai trong mã C #:

async void ListViewItemSelected(object sender, SelectedItemChangedEventArgs e)

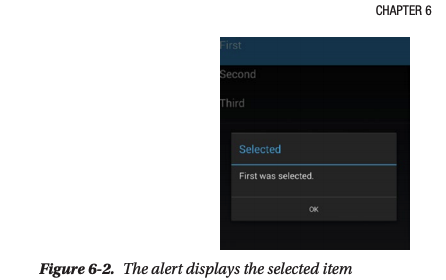
{

string item = (string)e.SelectedItem;

await DisplayAlert("Selected", item + " was selected.", "OK");

}

Sử dụng async / await không bắt buộc đối với các trình xử lý sự kiện này, nhưng đó là một thói quen tốt khi nào xử lý được thực hiện, để tránh thread UI. Sử dụng ItemTapped hoặc ItemSelected để chọn mục đầu tiên trong hình [6-2](https://translate.googleusercontent.com/translate_f#24).



Xóa hàng đã chọn (xóa phần tô sáng hàng) bằng cách đặt SelectedItem của ListView là null. Nơi dễ nhất để làm điều này là bên trong trình xử lý ItemTapped:

async voidListViewItemTapped (object sender, ItemTappedEventArgs e)

{

string item = (string) e.Item;

await DisplayAlert ( "Tapped", item+ "was selected.", "OK");

**((ListView)sender).SelectedItem = null;**

}

Nếu bạn đang sử dụng sự kiện ItemSelected, hãy lưu ý rằng việc thay đổi mục chọn sẽ kích hoạt lại sự kiện ItemSelected. Để xóa hàng đã chọn một cách an toàn, bạn cần kiểm tra xem e.SelectedItem có null hay không trước khi phản hồi sự kiện:

async void ListViewItemSelected (object sender, ItemTappedEventArgs e)

{

**if (e.SelectedItem == null) return;**

string item = (string)e.Item;

await DisplayAlert("Selected", item + " was selected.", "OK"); **((ListView)sender).SelectedItem = null;**

}

Ví dụ ListView để liên kết với danh sách các chuỗi, chọn một mục bằng cách sử dụng async / await để tạo nền cho trình xử lý sự kiện ItemTapped, sau đó xóa hàng đã chọn khi bạn hoàn thành

***6-2.****Tạo một*liên kết với một danh sách các chuỗi trong ListViewStrings.xaml



***Liệt kê 6-3.***Liên kết với một danh sách các chuỗi trong ListViewStrings.cs

public partial class ListViewStrings : ContentPage

{

public ListViewStrings()

{

InitializeComponent();

List Items = new List { "First", "Second", "Third" }; BindingContext = Items;

}

async void ListViewItemTapped (object sender, ItemTappedEventArgs e) {

string item = (string)e.Item;

await DisplayAlert("Tapped", item + " was selected.", "OK"); ((ListView)sender).SelectedItem = null;

}

async void ListViewItemSelected(object sender, SelectedItemChangedEventArgs e)

{

string item = (string)e.SelectedItem;

await DisplayAlert("Selected", item + " was selected.", "OK");

}

}

Liên kết với một mudel dữ liệu

Liên kết ListView với mô hình dữ liệu được thực hiện trong Xamarin.Forms thông qua việc sử dụng bộ tích hợp của ListView có tên là ItemTemplate.

Chuẩn bị mô hình dữ liệu và gán nó cho thuộc tính ListView.ItemsSource.

Sau đó liên kết từng thuộc tính của mô hình vào danh sách bằng ItemTemplate.SetBinding

Tạo một mô hình dữ liệu hoặc lớp tùy chỉnh, chứa các mục danh sách. Gọi là ListItem:

public class ListItem {

public string Title { get; set; }

public string Description { get; set; }

}

Tạo một danh sách ListItem :

List<ListItem> ListItems = new List<ListItem>

{

new ListItem {Title = "First", Description="1st item"},

new ListItem {Title = "Second", Description="2nd item"},

new ListItem {Title = "Third", Description="3rd item"}

};

Trỏ thuộc tính ItemSource của ListView vào ListItems. Sử dụng tên

DataModelList cho ListView mà bạn sẽ khai báo trong XAML:

DataModelList.DataModelList.ItemsSource = ListItems;

Đây là một cách thay thế để đặt ItemSource thành "{Binding.}" Và liên kết

ListItems cho ListView.

Tiếp theo, tạo ListView trong XAML và đặt tên là DataModelList. Định dạng danh sách hang sử dụng thuộc tính ItemTemplate của ListView. Tạo một lớp DataTemplate và sử dụng TextCell để hiển thị, TextCell sẽ hiển thị một tiêu đề cho mỗi hàng cộng với một số văn bản chi tiết mà bạn sẽ thêm trong một phút. Chỉ định thuộc tính để hiển thị là hàng chính văn bản bằng cách ràng buộc nó với TextCell.Text :

<ListView x:Name="DataModelList" ItemTapped="ListViewItemTapped" >

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding Title}"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

Điều này sẽ hiển thị cùng một danh sách nhưng từ một danh sách các lớp tùy chỉnh ListItem thay vì một danh sách các chuỗi (Hình  [6-3](https://translate.googleusercontent.com/translate_f#27) ).

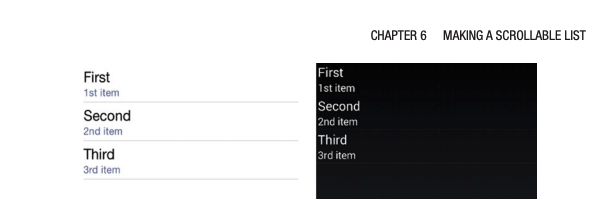


***Hình 6-3.****ListView này trông giống như Hình*[*6-1*](https://translate.googleusercontent.com/translate_f#22)*nhưng được điều khiển bởi ListItem*

Thêm một dòng văn bản mô tả cho mỗi hàng bằng cách ràng buộc thuộc tính TextCell:

<TextCell Text="{Binding Title}" **Detail="{Binding Description}"**/>

Điều này liên kết thuộc tính Description của lớp ListItem với thuộc tính Detail của TextCell. Hình  [6-4](https://translate.googleusercontent.com/translate_f#28) cho thấy kết quả.



Tip: màu phông chữ của TextCell có thể được đặt bằng thuộc tính TextColor của nó

<TextCell Text="{Binding Title}" Detail="{Binding Description}" **TextColor="Blue"**/>

Màu văn bản chi tiết (Description trong ví dụ này) có thể được đặt bằng cách sử dụng DetailColor:

<TextCell Text="{Binding Title}" Detail="{Binding Description}" **DetailColor="Red"**/>

Khi xử lý lựa chọn mục, hãy nhớ sử dụng mô hình dữ liệu ListItem:

async trống ListViewItemTapped (object sender, ItemTappedEventArgs e)

{

Mục ListItem = (ListItem) e.Item;

đang chờ DisplayAlert ("Khai thác", item.Title + "đã được chọn.", "OK");

Người gửi ((ListView)) .SelectedItem = null;

}

async void ListViewItemTapped (object sender, ItemTappedEventArgs e)

{

**ListItem item = (ListItem)e.Item;**

await DisplayAlert("Tapped", **item.Title** + " was selected.", "OK");

((ListView)sender).SelectedItem = null;

}

**CODE COMPLETE: Binding to a Data Model**

6.1 Tạo ListViewDataModel.xaml

<?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="chapter6.ListViewDataModel">

<ContentPage.Padding>

<OnPlatform x:TypeArguments="Thickness">

<On Platform="Android">10,0,10,5</On>

<On Platform="iOS">10,20,10,5</On>

</OnPlatform> </ContentPage.Padding>

<ListView x:Name="DataModelList" ItemTapped="ListViewItemTapped" >

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding

Title}" Detail="{Binding Description}"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ContentPage>

Với file cs như sau:

using System;

using System.Collections.Generic;

using Xamarin.Forms;

namespace chapter6

{

public partial class ListViewDataModel : ContentPage

{

public ListViewDataModel()

{

InitializeComponent();

List<ListItem> ListItems = new List<ListItem>

{

new ListItem { Title = "Khoa", Description="1st item"},

new ListItem { Title = "Duong", Description="2nd item"},

new ListItem Title = "Tai", Description="3rd item"}

};

DataModelList.ItemsSource = ListItems;

}

async void ListViewItemTapped(object sender, ItemTappedEventArgs e)

{

ListItem item = (ListItem)e.Item;

await DisplayAlert("Tapped", item.Title + " was selected.", "OK");

((ListView)sender).SelectedItem = null;

}

public class ListItem

{

public string Title { get; set; }

public string Description { get; set; }

}

}

}

**Thêm một ảnh vô ListView**

Ta thay thế TextCell bằng ImageCell và thêm Source cho ImageSource, Ta sẽ có code như sau :

<?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="chapter6.ListViewDataModel">

<ContentPage.Padding>

<OnPlatform x:TypeArguments="Thickness">

<On Platform="Android">10,0,10,5</On>

<On Platform="iOS">10,20,10,5</On>

</OnPlatform> </ContentPage.Padding>

<ListView x:Name="DataModelList" ItemTapped="ListViewItemTapped" >

<ListView.ItemTemplate>

<DataTemplate>

<ImageCell ImageSource="{Binding Source}" Text="{Binding

Title}" Detail="{Binding Description}"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ContentPage>

File CS tương ứng :

using System;

using System.Collections.Generic;

using Xamarin.Forms;

namespace chapter6

{

public partial class ListViewDataModel : ContentPage

{

public ListViewDataModel()

{

InitializeComponent();

List<ListItem> ListItems = new List<ListItem>

{

new ListItem { Source ="imageSmall.png", Title = "Khoa", Description="1st item"},

new ListItem { Source ="imageSmall.png", Title = "Duong", Description="2nd item"},

new ListItem { Source ="imageSmall.png", Title = "Tai", Description="3rd item"}

};

DataModelList.ItemsSource = ListItems;

}

async void ListViewItemTapped(object sender, ItemTappedEventArgs e)

{

ListItem item = (ListItem)e.Item;

await DisplayAlert("Tapped", item.Title + " was selected.", "OK");

((ListView)sender).SelectedItem = null;

}

public class ListItem

{

public string Title { get; set; }

public string Description { get; set; }

public string Source { get; set; }

}

}

}

**Tạo một List tuỳ chỉnh**

Tuỳ chỉnh bố cục một danh sách bằng cách tạo một lớp kế thừa từ ViewCell. Sau đó chúng ta tạo bố cục tuỳ chỉnh cho danh sách:

Khai báo các đối tượng cần thiết trong file cs :

public class ListItem

{

public string Source { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string Price { get; set; }

}

Chuyền dữ liệu vào list:

List<ListItem> ListItems = new List<ListItem>

{

new ListItem {Title = "First", Description="1st item", Price="$100.00"},

new ListItem {Title = "Second", Description="2nd item", Price="$200.00"},

new ListItem {Title = "Third", Description="3rd item", Price="$300.00"}

};

Tiếp theo chúng ta thiết kế Layout tuỳ chỉnh cho List:

<ListView x:Name="CustomList">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout HorizontalOptions="StartAndExpand"

Orientation="Horizontal" Padding="25,10,55,15">

<StackLayout HorizontalOptions="StartAndExpand"

Orientation="Vertical">

<Label HorizontalOptions="Start" FontSize="20"

FontAttributes="Bold" TextColor="White"

Text="{Binding Title}"/>

<Label HorizontalOptions="Start" FontSize="12"

FontAttributes="Bold" TextColor="White"

Text="{Binding Description}"/>

</StackLayout>

<Label HorizontalOptions="End" FontSize="25"

TextColor="Aqua" Text="{Binding Price}"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

Lưu ý đoạn code trên ta sử dụn 2 StackLayout để tạo một bố ngang trong có chia làm bố cục dọc cho layout của chúng ta.

Ngoài ra bạn có thể tuỳ chỉnh listview với RowHeight và BackgroudColor, ví dụ:

**<ListView x: Name = "CustomList" RowHeight = "80" BackgroundColor = "Black">**

Ta có mã file cs tương ướng:

using System;

using System.Collections.Generic;

using Xamarin.Forms;

namespace chapter6

{

public partial class ListViewCustom : ContentPage

{

public ListViewCustom()

{

InitializeComponent();

List<ListItem> ListItems = new List<ListItem>

{

new ListItem {Title = "Khoa", Description="1st item",

Price="$100.00"},

new ListItem {Title = "Dương", Description="2nd item",

Price="$200.00"},

new ListItem {Title = "Tài", Description="3rd item",

Price="$300.00"}

};

CustomList.ItemsSource = ListItems;

}

async void ListViewItemTapped(object sender, ItemTappedEventArgs e)

{

ListItem item = (ListItem)e.Item;

await DisplayAlert("Tapped", item.Title + " was selected.", "OK");

((ListView)sender).SelectedItem = null;

}

public class ListItem

{

public string Source { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string Price { get; set; }

}

}

}

**Thêm nút cho listview**

Ta có cú pháp thêm nút như sau :

<Button x:Name="BuyButton" Text="Buy Now" BackgroundColor="Teal"

HorizontalOptions="EndAndExpand" Clicked="BuyClicked"

CommandParameter="{Binding .}"/>

Hàm tương ứng :

public async void BuyClicked(object sender, EventArgs e)

{

var b = (Button)sender;

var item = (ListItem)b.CommandParameter;

await DisplayAlert("Clicked", item.Title.ToString() + " button was

clicked", "OK");

}

Thêm CommandParameter vào ViewCell để xách định vị trí được .

<Button x:Name="BuyButton" Text="Buy Now" BackgroundColor="Teal"

HorizontalOptions="EndAndExpand" Clicked="BuyClicked"

**CommandParameter**="{Binding .}"/>

Hàm xử lý sự kiện :

public async void BuyClicked(object sender, EventArgs e)

{

var b = (Button)sender;

var item = (ListItem)b.CommandParameter;

await DisplayAlert("Clicked", item.Title.ToString() + " button was

clicked", "OK");

}

Thực hiện tạo file ListViewButton.xaml :

<?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="chapter6.ListViewButton">

<ContentPage.Padding>

<OnPlatform x:TypeArguments="Thickness">

<On Platform="iOS">10,20,10,5</On>

<On Platform="Android">10,0,10,5</On>

</OnPlatform>

</ContentPage.Padding>

<ListView x:Name="ButtonList" RowHeight="100" BackgroundColor="Black"

HasUnevenRows="true" >

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout HorizontalOptions="StartAndExpand"

Orientation="Horizontal" Padding="5,10,5,15">

<StackLayout HorizontalOptions="Start"

Orientation="Vertical">

<Label HorizontalOptions="Start" FontSize="20"

FontAttributes="Bold" TextColor="White"

Text="{Binding Title}"/>

<Label HorizontalOptions="Start" FontSize="12"

FontAttributes="Bold" TextColor="White"

Text="{Binding Description}"/>

</StackLayout>

<StackLayout HorizontalOptions="EndAndExpand"

Orientation="Horizontal" WidthRequest="260">

<Label HorizontalOptions="Center" FontSize="25"

TextColor="Aqua" Text="{Binding Price}"/>

<Button x:Name="BuyButton" Text="Buy Now"

BackgroundColor="Teal" HorizontalOptions=

"EndAndExpand" Clicked="BuyClicked"

CommandParameter="{Binding .}"/>

</StackLayout>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ContentPage>

Với file cs tương ứng:

using System;

using System.Collections.Generic;

using Xamarin.Forms;

namespace chapter6

{

public partial class ListViewButton : ContentPage

{

public ListViewButton()

{

InitializeComponent();

List<ListItem> ListItems = new List<ListItem> {

new ListItem {Title = "First", Description="1st item",

Price="$100.00"},

new ListItem {Title = "Second", Description="2nd item",

Price="$200.00"},

new ListItem {Title = "Third", Description="3rd item",

Price="$300.00"}

};

ButtonList.ItemsSource = ListItems;

}

public async void BuyClicked(object sender, EventArgs e)

{

var b = (Button)sender;

var item = (ListItem)b.CommandParameter;

await DisplayAlert("Clicked", item.Title.ToString() + " button was clicked", "OK");

}

public class ListItem

{

public string Title { get; set; }

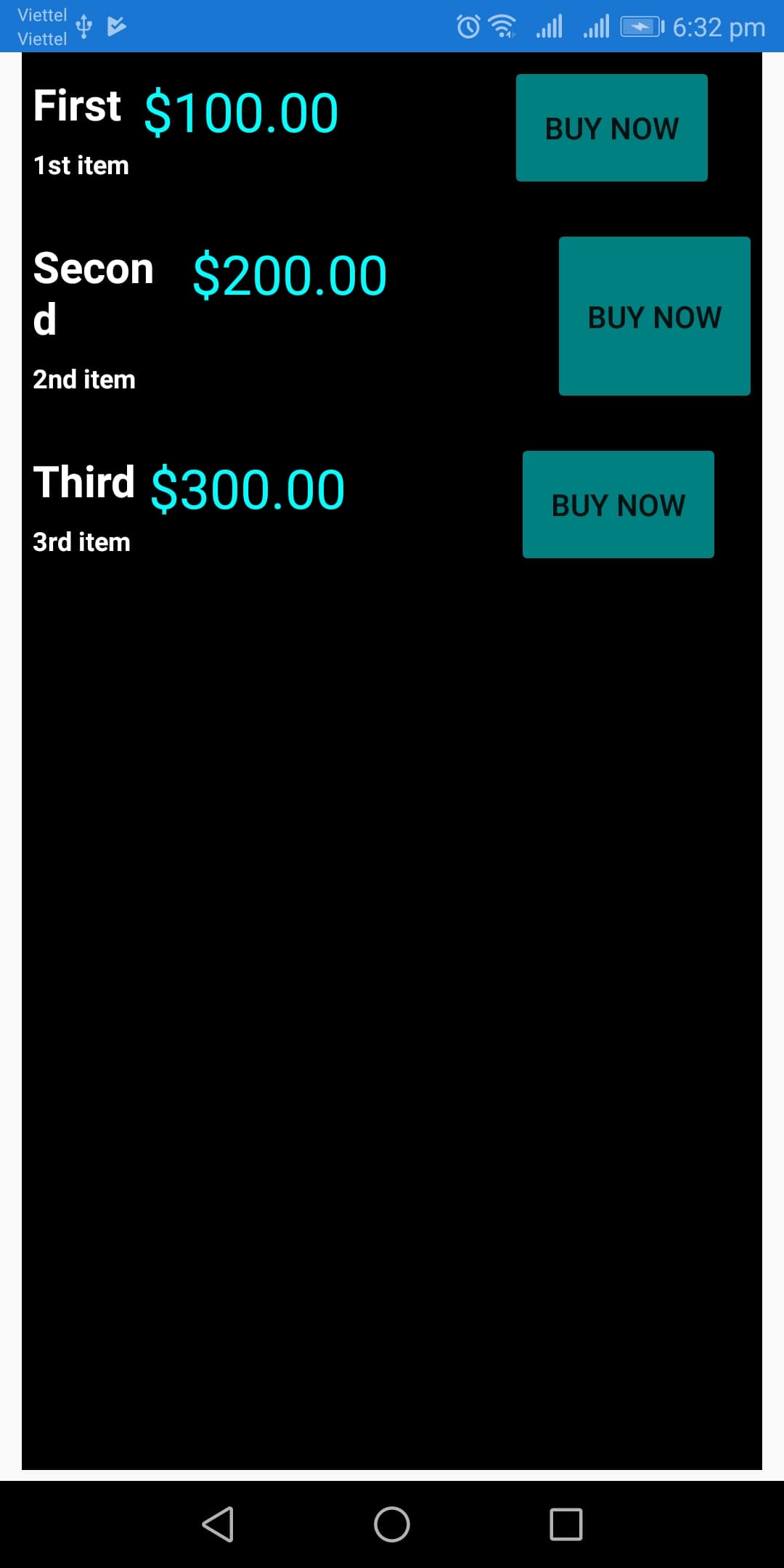
public string Description { get; set; }

public string Price { get; set; }

}

}

}



**Using Context Actions**

Chúng ta sử dụng MenuItem đặt trong ViewCell để tuỳ chỉnh danh sách. Sau đó tạo một MenuItem để hiển thị thay thế cho nút Botton. Chúng ta sẽ có mã code như sau:

<ViewCell.ContextActions>

<**MenuItem** Text="More" **Clicked**="MoreClicked"

CommandParameter="{Binding .}"/>

<**MenuItem** Text="Delete" **Clicked**="DeleteClicked"

CommandParameter="{Binding .}" IsDestructive="True" />

</ViewCell.ContextActions>

Với mã xử lý cs:

public async void MoreClicked(object sender, EventArgs e)

{

var mi = ((MenuItem)sender);

var item = (ListItem)(mi.CommandParameter);

await DisplayAlert("Clicked", item.Title.ToString() + " More button

was clicked", "OK");

}

Thực hiện:

Tạo một file ListViewContextAction.xaml :

<?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="chapter6.ListViewContextAction">

<ContentPage.Padding>

<OnPlatform x:TypeArguments="Thickness">

<On Platform="iOS">10,20,10,5</On>

<On Platform="Android">10,0,10,5</On>

</OnPlatform>

</ContentPage.Padding>

<ListView x:Name="ContextList" RowHeight="100" BackgroundColor="Black"

HasUnevenRows="true">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<ViewCell.ContextActions>

<MenuItem Text="More" Clicked="MoreClicked"

CommandParameter="{Binding .}"/>

<MenuItem Text="Delete" Clicked="DeleteClicked"

CommandParameter="{Binding .}" IsDestructive="True" />

</ViewCell.ContextActions>

<StackLayout HorizontalOptions="StartAndExpand"

Orientation="Horizontal" Padding="5,10,5,15">

<StackLayout HorizontalOptions="StartAndExpand"

Orientation="Vertical">

<Label HorizontalOptions="Start" FontSize="20"

FontAttributes="Bold" TextColor="White"

Text="{Binding Title}"/>

<Label HorizontalOptions="Start" FontSize="12"

FontAttributes="Bold" TextColor="White"

Text="{Binding Description}"/>

</StackLayout>

<Label HorizontalOptions="End" FontSize="25"

TextColor="Aqua" Text="{Binding Price}"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ContentPage>

Với đoạn mã cs tương ứng:

using System;

using System.Collections.Generic;

using Xamarin.Forms;

namespace chapter6

{

public partial class ListViewContextAction : ContentPage

{

public ListViewContextAction()

{

InitializeComponent();

InitializeComponent();

List<ListItem> ListItems = new List<ListItem> {

new ListItem {Title = "First", Description="1st item",

Price="$100.00"},

new ListItem {Title = "Second", Description="2nd item",

Price="$200.00"},

new ListItem {Title = "Third", Description="3rd item",

Price="$300.00"}

};

ContextList.ItemsSource = ListItems;

}

public async void MoreClicked(object sender, EventArgs e)

{

var mi = ((MenuItem)sender);

var item = (ListItem)(mi.CommandParameter);

await DisplayAlert("Clicked", item.Title.ToString() + " More button was clicked", "OK");

}

public async void DeleteClicked(object sender, EventArgs e)

{

var mi = ((MenuItem)sender);

var item = (ListItem)mi.CommandParameter;

await DisplayAlert("Clicked", item.Title.ToString() + " Delete button was clicked", "OK");

}

public class ListItem

{

public string Title { get; set; }

public string Description { get; set; }

public string Price { get; set; }

}

}

}

**Grouping Headers**

Tạo một nhóm với code :

List<Group> itemsGrouped = new List<Group> {

new Group ("Important", new List<ListItem>{

new ListItem {Title = "First", Description="1st item"},

new ListItem {Title = "Second", Description="2nd item"},

}),

new Group ("Less Important", new List<ListItem>{

new ListItem {Title = "Third", Description="3rd item"}

})

};

Tạo một ListVieu với tên là GroupedList :

<ListView x:Name="GroupedList" IsGroupingEnabled="true"

GroupDisplayBinding="{Binding Key}">

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding Title}" Detail="{Binding

Description}"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

Thêm vào constructor :

GroupedList.ItemsSource = itemsGrouped;

Thực hiện :

Tạo file ListViewGrouped.xaml :

<?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="chapter6.ListViewGrouped">

<ContentPage.Padding>

<OnPlatform x:TypeArguments="Thickness">

<On Platform="iOS">10,20,10,5</On>

<On Platform="Android">10,0,10,5</On>

</OnPlatform>

</ContentPage.Padding>

<ListView x:Name="GroupedList" IsGroupingEnabled="true"

GroupDisplayBinding="{Binding Key}">

<ListView.ItemTemplate>

<DataTemplate>

<TextCell Text="{Binding Title}" Detail="{Binding

Description}"/>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</ContentPage>

Nội dung file cs tương ứng:

using System;

using System.Collections.Generic;

using Xamarin.Forms;

using static chapter6.ListViewGrouped.Group;

namespace chapter6

{

public partial class ListViewGrouped : ContentPage

{

public ListViewGrouped()

{

List<Group> itemsGrouped;

InitializeComponent();

itemsGrouped = new List<Group> {

new Group("Important", new List<ListItem>

{

new ListItem {Title = "First", Description="1st item"},

new ListItem {Title = "Second", Description="2nd item"}

}),

new Group("Less Important", new List<ListItem>

{

new ListItem {Title = "Third", Description="3rd item"}

})

};

GroupedList.ItemsSource = itemsGrouped;

}

public class Group : List<ListItem>

{

public string Key

{

get;

private set;

}

public Group(string key, List<ListItem> listItems)

{

Key = key;

foreach (var item in listItems)

this.Add(item);

}

public class ListItem

{

public string Title { get; set; }

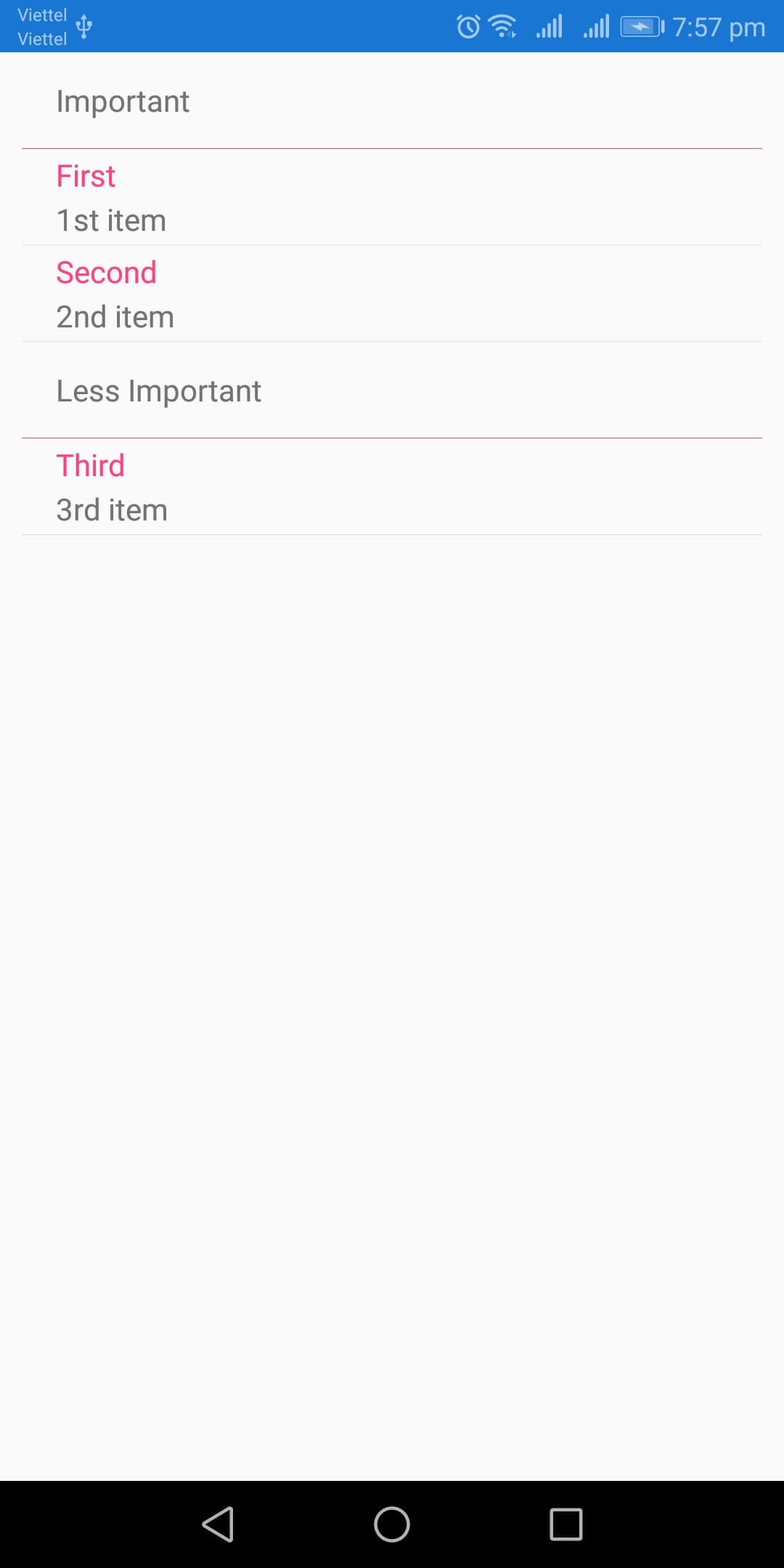
public string Description { get; set; }

}

}

}

}



**Creating a Jump List**

Danh sách dài có thể khó sử dụng và yêu cầu cuộn nhanh bằng cách sử dụng danh sách nhảy, đó là danh sách c ác phím bên phải cho phép di chuyển nhanh qua danh sách.

Gán các giá trị danh sách nhảy bằng cách ràng buộc thuộc tính trong mô hình nhóm với thuộc tính ListView.groupShortNameBinding.

<ListView x:Name="JumpList" IsGroupingEnabled="true" HasUnevenRows="true"

**GroupShortNameBinding** = "{Binding Key}">