Работа со списками и коллекциями данных

Во время работы с пользовательским интерфейсом часто приходится иметь дело со списками и коллекциями данных различного типа, требующими выполнения прокрутки вследствие того, что элементы списков и данные невозможно отобразить в полном объеме в пределах пользовательского интерфейса.

Для прокрутки списков и коллекций могут быть использованы элементы управления пользовательского интерфейса, относящиеся в Xamarin.Forms к визуальным элементам для отображения коллекций (CarouselView, IndicatorView, CollectionView, ListView, TableView).

Рассматриваются возможности работы мобильного программного приложения по работе с прокручиваемым списком элементов данных с помощью элемента управления CarouselView.

По умолчанию элементы данных в элементе управления CarouselView отображаются по горизонтали. При этом, в текущий момент времени в пользовательском интерфейсе будет доступен для просмотра только один элемент данных. К элементу управления можно применить жест прокрутки в прямом и обратном направлении для просмотра недоступных для просмотра элементов данных.

Элемент управления CarouselView имеет следующие свойства:

свойство ItemsSource задает коллекцию элементов для отображения и имеет значение по умолчанию null;

свойство ItemTemplate указывает на шаблон, применяемый к каждому отображаемому элементу коллекции;

свойство ItemsLayout – представляет собой класс, который задает макет для отображения элемента управления;

свойство PeekAreaInsets - указывает, сколько элементов можно сделать смежными и частично видимыми для пользователя.

Класс ItemsLayout имеет следующие свойства:

свойство Orientation задает направление, в котором в элемент управления CarouselView добавляются новые элементы данных в отображаемую коллекцию (значение свойства Vertical указывает, что коллекция в элементе управления CarouselView будет расширяться по вертикали по мере добавления элементов данных, а значение Horizontal указывает, что коллекция в элементе управления CarouselView будет расширяться по горизонтали по мере добавления элементов);

свойство SnapPointsAlignment задает способ выравнивания точек привязки элементов данных коллекции;

свойство SnapPointsType задает поведение точек привязки элементов данных при прокрутке коллекции;

свойство ItemSpacing задает пустое пространство вокруг каждого элемента данных в коллекции, при этом значение этого свойства по умолчанию равно нулю.

Указанное выше свойство SnapPointsType может иметь следующие значения:

значение None указывает на то, что при прокрутке коллекции не происходит привязки к элементам коллекции;

значение Mandatory указывает на то, что элементы при прокрутке привязываются к ближайшей точке привязки, в которой будет останавливаться прокрутка;

значение MandatorySingle указывает на то, что прокрутка коллекции производится по одному элементу.

Указанное выше свойство SnapPointsAlignment класса ItemsLayout может принимать следующие значения:

значение Start является значением по умолчанию и указывает на то, что первый элемент коллекции будет находиться в верхней части в элементе управления CarouselView;

значение Center указывает на то, что в верхней части в элементе управления CarouselView отображается соответственно только нижняя половина первого элемента коллекции;

значение End указывает, что последний элемент коллекции будет находиться в нижней части в элементе управления CarouselView.

Также элемент управления CarouselView имеет свойства, управляющие взаимодействием с пользователем:

свойство CurrentItem задает текущий отображаемый элемент;

свойство CurrentItemChangedCommand задает команду, которая выполняется при изменении элемента данных коллекции, отображаемой в пользовательском интерфейсе в текущий момент времени;

свойство CurrentItemChangedCommandParameter задает параметр, который передается в команду, указанную в свойстве CurrentItemChangedCommand;

свойство IsBounceEnabled определяет, будет ли элемент управления CarouselView перематывать коллекцию при достижении ее границы, при этом значение этого свойства по умолчанию равно true;

свойство IsSwipeEnabled, значение которого по умолчанию равно true, определяет, будет ли жест прокрутки изменять отображаемый элемент коллекции (прокрутку элементов коллекции можно отключить, задав свойству значение false);

свойство Position задает индекс текущего отображаемого элемента коллекции;

свойство PositionChangedCommand задает команду, которая выполняется при изменении расположения элемента коллекции;

свойство PositionChangedCommandParameter задает параметр, который передается в команду, указанную в свойстве PositionChangedCommand;

свойство VisibleViews является свойством только для чтения и содержит элементы, видимые в данный момент.

Элемент управления CarouselView заполняется данными путем присваивания коллекции данных свойству ItemsSource.

Далее приведены фрагменты программного кода XAML и Си Шарп с примерами присвоения коллекции данных Rockets свойству ItemsSource.

В приведенном ниже программном коде XAML присвоение коллекции данных производится с помощью расширения разметки Binding.

```
<CarouselView ItemsSource="{Binding Rockets}" />
```

В программном коде Си Шарп присвоение коллекции данных производится с помощью метода SetBinding.

```
CarouselView carouselView = new CarouselView();
carouselView.SetBinding
                (ItemsView.ItemsSourceProperty, "Rockets");
```

Внешний вид каждого элемента коллекции в элементе управления CarouselView может быть определен путем присвоения свойству ItemTemplate шаблона данных DataTemplate.

Элементы, указанные в шаблоне данных DataTemplate, определяют единый внешний вид каждого элемента коллекции данных, отображаемой с помощью элемента управления CarouselView.

Далее приведен фрагмент программного кода XAML, в котором рассматривается использование шаблона DataTemplate для каждого элемента коллекции данных. В состав шаблона элемента данных входит элемент управления Image, в который загружается изображение, путь к которому указывается в переменной ImageUrl. Загрузка элементов данных производится из коллекции Rockets.

```
<CarouselView ItemsSource="{Binding Rockets}">
  <CarouselView.ItemTemplate>
    <DataTemplate>
          <Image Source="{Binding ImageUrl}"
              Aspect="AspectFill"
              HeightRequest="100"
              WidthRequest="100"
              HorizontalOptions="Center" />
    </DataTemplate>
  </CarouselView.ItemTemplate>
</CarouselView>
```

Внешний вид каждого элемента данных коллекции, отображаемой с помощью элемента управления CarouselView также может быть выбран из списка шаблонов. Выбранному шаблону соответствует объект DataTemplateSelector, который присваивается свойству ItemTemplate для элемента данных коллекции.

Далее приведен фрагмент программного кода XAML, в котором класс RocketTemplateSelector с ключом "RS" имеет два свойства Rocket1 и Rocket2, для которых установлены шаблоны элементов данных Rocket1Template и Rocket2Template. Свойству ItemTemplate элемента управления CarouselView присваивается список шаблонов с помощью ключа RS.

```
  <ContentPage.Resources>
    <DataTemplate x:Key="Rocket1Template">

    </DataTemplate>
```

```
        <DataTemplate x:Key="Rocket2Template">

        </DataTemplate>
        <controls:RocketTemplateSelector x:Key="RS"
            Rocket1="{StaticResource Rocket1Template}"
            Rocket2="{StaticResource Rocket2Template }" />
    </ContentPage.Resources>
    <CarouselView ItemsSource="{Binding Rockets}"
                  ItemTemplate="{StaticResource RS}" />
```

В случае выбора шаблона из списка, задаваемого классом RocketTemplateSelector, происходит срабатывание обработчика OnSelectTemplate.

Далее приведен фрагмент программного кода Си Шарп для класса RocketTemplateSelector, являющегося дочерним классом от класса DataTemplateSelector. В классе RocketTemplateSelector объявлены свойства Rocket1 и Rocket2, а также обработчик OnSelectTemplate.

```
public class RocketTemplateSelector: DataTemplateSelector
{
    public DataTemplate Rocket1 { get; set; }
    public DataTemplate Rocket2 { get; set; }

    protected override DataTemplate OnSelectTemplate
                        (object item, BindableObject container)
    {
        // Операторы для обработки выбора шаблона
    }
}
```

Элементу управления CarouselView соответствует событие CurrentItemChanged, которое возникает при изменении значения свойства CurrentItem.

Объект CurrentItemChangedEventArgs, соответствующий событию CurrentItemChanged, имеет два свойства:

свойство PreviousItem задает предыдущий элемент после изменения отображаемого элемента коллекции;

свойство CurrentItem задает текущий отображаемый элемент коллекции.

Также элементу управления CarouselView соответствует событие PositionChanged, которое возникает при изменении значения свойства Position.

Объект PositionChangedEventArgs, соответствующий событию PositionChanged, имеет два свойства:

свойство PreviousPosition задает предыдущее расположение отображаемого элемента коллекции после изменения значения свойства Position;

свойство CurrentPosition задает текущую позицию элемента коллекции после изменения значения свойства Position.

При смене отображаемого в данный момент элемента коллекции свойству CurrentItem будет присвоен вновь отображаемый элемент коллекции. При изменении значения свойства CurrentItem будет выполняться команда CurrentItemChangedCommand с параметром CurrentItemChangedCommandParameter. Затем обновляется значение свойства Position и вызывается событие CurrentItemChanged. При изменении значения свойства Position происходит выполнение команды PositionChangedCommand и срабатыванию события PositionChanged.

Далее приведен фрагмент программного кода XAML, в котором показан элемент управления CarouselView, использующий обработчик событий OnCurrentItemChanged для реагирования на изменение элемента коллекции Rockets, отображаемого в текущий момент времени.

```
<CarouselView ItemsSource="{Binding Rockets}"
        CurrentItemChanged="OnCurrentItemChanged">
</CarouselView>
```

Также в программном коде Си Шарп показан обработчик OnCurrentItemChanged, на который ссылается программный код XAML и который выполняется при срабатывании события CurrentItemChanged.

```
void OnCurrentItemChanged
            (object sender, CurrentItemChangedEventArgs e)
{
   Rocket previousItem = e.PreviousItem as Rocket;
   Rocket currentItem = e.CurrentItem as Rocket;
}
```

Далее приведен фрагмент программного кода, в котором рассмотрен элемент управления CarouselView, использующий команду для реагирования на изменение элемента коллекции Rockets, отображаемого в текущий момент времени. Свойство CurrentItemChangedCommand связывается с командой ItemChangedCommand, передавая ей значение свойства CurrentItem в качестве аргумента.

```
<CarouselView ItemsSource="{Binding Rockets}"
        CurrentItemChangedCommand=
                "{Binding ItemChangedCommand}"
        CurrentItemChangedCommandParameter=
                "{Binding Source={RelativeSource Self},
                  Path=CurrentItem}">
   ...
</CarouselView>
```

Команда ItemChangedCommand в итоге может реагировать на изменение элемента коллекции Rockets, отображаемого в текущий момент времени.

```
public ICommand ItemChangedCommand
            => new Command<Rocket>((item) =>
{
   PreviousRocket = CurrentRocket;
   CurrentRocket = item;
```

```
});
```

При изменении отображаемого в текущий момент элемента коллекции значению свойства Position будет присвоен индекс отображаемого элемента. При изменении значения свойства Position выполняется команда PositionChangedCommand, в которую передается параметр PositionChangedCommandParameter и затем происходит событие PositionChanged.

Далее приводится фрагменты программного кода XAML и Си Шарп, в которых приведен пример использования обработчика OnPositionChanged для события PositionChanged, а также пример обращения к команде PositionChangedCommand, выполняемой в случае срабатывания этого события, а также определения параметра PositionChangedCommandParameter, который передается команде PositionChangedCommand.

```
<CarouselView ItemsSource="{Binding Rockets}"
    PositionChanged="OnPositionChanged"
    PositionChangedCommand=
                "{Binding PositionChangedCommand}"
    PositionChangedCommandParameter=
          "{Binding Source={RelativeSource Self},
            Path=Position}">
  ...
</CarouselView>
```

В программном коде Си Шарп приведен обработчик события PositionChanged, на который ссылается приведенный ранее программный код XAML. При выполнении команды, в которую передается параметр Position в качестве аргумента. После этого производится изменения значений свойств PreviousPosition и CurrentPosition.

```
public ICommand PositionChangedCommand =>
                    new Command<int>((position) =>
{
  PreviousPosition = CurrentPosition;
  CurrentPosition = position;
});
```

Элемент управления CarouselView определяет четыре визуальных состояния, которые могут быть использованы для инициации визуальных изменений элементов отображаемой коллекции:

состояние CurrentItem представляет визуальное состояние для элемента коллекции, отображаемого в данный момент времени;

состояние PreviousItem представляет визуальное состояние для предыдущего элемента коллекции;

состояние NextItem представляет визуальное состояние для следующего элемента коллекции;

состояние DefaultItem представляет визуальное состояние для остальных элементов коллекции.

Далее приведен фрагмент программного кода, в котором рассмотрен пример задания визуального состояния CurrentItem. Визуальное состояние указывает, что для элемента коллекции, отображаемого в текущий момент времени в CarouselView, свойству FontSize присвоено значение, равное 30.

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="CommonStates">
        <VisualState x:Name="CurrentItem">
            <VisualState.Setters>
                <Setter Property="FontSize"
                        Value="30" />
            </VisualState.Setters>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

Элемент управления CarouselView имеет следующие свойства, связанные с использованием полос прокрутки:

свойство HorizontalScrollBarVisibility указывает способ отображения горизонтальной полосы прокрутки для коллекции;

свойство VerticalScrollBarVisibility указывает на способ отображения вертикальной полосы прокрутки для коллекции;

свойство IsDragging по умолчанию равно false и указывает, выполняется ли в настоящий прокрутка элемента управления CarouselView;

свойство IsScrollAnimated по умолчанию равно true и указывает, будет ли выполняться анимация при перемещении между элементами управления CarouselView с использованием полос прокрутки (при установке значения false анимация будет отключена для обеих полос прокрутки);

свойство ItemsUpdatingScrollMode представляет поведение прокрутки CarouselView при добавлении новых элементов к нему.

Свойства HorizontalScrollBarVisibility и VerticalScrollBarVisibility могут иметь следующие значения:

значение Default является значением по умолчанию и указывает на поведение полосы прокрутки для платформы, на которой запущено программное приложение;

значение Always указывает на то, что полосы прокрутки будут видимы, даже если содержимое коллекции умещается в пользовательском интерфейсе;

значение Never указывает на то, что полосы прокрутки не будут видимы, даже если содержимое коллекции не умещается в пользовательском интерфейсе.

Свойство ItemsUpdatingScrollMode может принимать следующие значения:

значение KeepItemsInView сохраняет первый элемент в коллекции при добавлении новых элементов, при этом данное значение принимается по умолчанию;

значение KeepScrollOffset показывает, что текущая позиция прокрутки сохраняется при добавлении новых элементов в коллекцию;

значение KeepLastItemInView показывает, что производится сохранение последнего элемента списка при добавлении новых элементов коллекции.

Элемент управления CarouselView имеет два вида метода ScrollTo. Первый вид метода ScrollTo выполняет прокрутку до элемента коллекции с указанным индексом. Второй вид метода ScrollTo выполняет прокрутку до элемента коллекции с заданным именем. При этом при прокрутке элемента коллекции точное его расположение после прокрутки можно указать с помощью одного из значений ScrollToPosition:

значения MakeVisible указывает на то, что элемент коллекции должен прокручиваться до тех пор, пока он не будет виден;

значение Start указывает на то, что элемент коллекции должен прокручиваться до начала области пользовательского интерфейса, в которой отображается коллекция;

значение Center указывает на то, что элемент коллекции должен прокручиваться до центра области пользовательского интерфейса, в которой отображается коллекция;

значение End указывает на то, что элемент коллекции должен прокручиваться до конца области пользовательского интерфейса, в которой отображается коллекция.

Далее приведены фрагменты программного кода Си Шарп, в котором показано, как с помощью элемента управления CarouselView прокручивать коллекцию до элемента с индексом 6. Показано также, что для аргумента animate метода ScrollTo можно установить значение false для отключения анимации при прокрутке до элемента с именем Rocket. Кроме этого, показано, как применять прокрутку до элемента коллекции с заданным именем с использованием различных значений ScrollToPosition.

```
carouselView.ScrollTo(6);
carouselView.ScrollTo(Rocket, animate: false);
carouselView.ScrollTo
            (Rocket, position: ScrollToPosition.MakeVisible);
carouselView.ScrollTo(Rocket, position: ScrollToPosition.Start);
carouselView.ScrollTo
                    (Rocket, position: ScrollToPosition.Center);
```

Событие ScrollToRequested возникает при прокрутке одной их полос прокрутки. Кроме этого, элементу управления CarouselView соответствует событие Scrolled, которое срабатывает в случае окончания прокрутки. Это событие следует использовать, если требуется получить данные о прокрутке.

Элемент управления CarouselView имеет также свойства, которые можно использовать, когда нет данных для отображения коллекции в элементе управления:

свойство EmptyView равно по умолчанию null и задает строку, ссылку или элемент управления, которые будут отображаться в случае, если свойство ItemsSource имеет значение null;

свойство EmptyViewTemplate имеет по умолчанию значение null и задает шаблон для форматирования объекта, присвоенного свойству EmptyView.

Далее приведен фрагмент программного кода, в котором рассматривается пример задания строки для свойства EmptyView в случае, если в коллекции Rockets нет элементов данных.

```
<CarouselView ItemsSource="{Binding Rockets}"
    EmptyView="Нет данных для отображения" />
```

Далее приведен фрагмент программного кода, в котором рассматривается пример задания шаблона элемента управления для свойства EmptyView. В качестве шаблона для отображения (в случае отсутствия элементов коллекции) использован элемент управления Label с настроенным размерами шрифта, стилем и положением текста.

```
<CarouselView ItemsSource="{Binding Rockets}">
    <CarouselView.EmptyView>
      … Форматируемый объект …
    </CarouselView.EmptyView>
    <CarouselView.EmptyViewTemplate>
      <DataTemplate>
        <Label Text ="Нет данных для отображения"
            FontAttributes="Bold" FontSize="18"
            HorizontalTextAlignment="Center" />
      </DataTemplate>
    </CarouselView.EmptyViewTemplate>
    <CarouselView.ItemTemplate>
      ...
    </CarouselView.ItemTemplate>
</CarouselView>
```

Совместно с элементом управления CarouselView может использоваться индикатор прокрутки IndicatorView для отображения количества элементов данных в коллекции или списке в CarouselView, а также для отображения текущей позиции элемента данных в списке или коллекции.

Элемент управления IndicatorView имеет следующие свойства:

свойство Count задает количество отображаемых индикаторов для коллекции;

свойство HideSingle по умолчанию равно true и указывает, должен ли индикатор быть скрытым, если он один;

свойство IndicatorColor задает цвет индикаторов;

свойство IndicatorSize задает размер индикаторов, при этом значение свойства по умолчанию равно 6;

свойство IndicatorLayout определяет класс макета, используемый для визуализации элемента управления;

свойство IndicatorTemplate задает шаблон, определяющий внешний вид индикатора;

свойство IndicatorsShape задает форму индикатора;

свойство ItemsSource задает коллекцию, для которой будут отображаться индикаторы, при этом значение свойства будет автоматически установлено при установке значения свойства CarouselView.IndicatorView;

свойство MaximumVisible задает максимальное количество видимых индикаторов;

свойство Position задает текущий индекс индикатора;

свойство SelectedIndicatorColor задает цвет индикатора, представляющий текущий элемент коллекции в CarouselView.

Далее приведен фрагмент программного кода XAML, который отображает элемент управления IndicatorView под элементом управления CarouselView для индикации положения каждого элемента данных из коллекции данных.

```
<CarouselView ItemsSource="{Binding Rockets}"
          IndicatorView="indView">
  <CarouselView.ItemTemplate>
    <!—Шаблон для элемента коллекции -->
  </CarouselView.ItemTemplate>
</CarouselView>
<IndicatorView x:Name="indView"
          IndicatorColor="LightGray"
          SelectedIndicatorColor="DarkGray"
          HorizontalOptions="Center" />
```

Заполнение элемента управления CarouselView производится за счет привязки свойства ItemsSource к коллекции данных Rockets. Каждый элемент коллекции Rockets отображается однообразно в соответствии с шаблоном. Элемент управления CarouselView связан с индикатором прокрутки IndicatorView с именем IndView. Элемент индикатора прокрутки с именем IndView, соответствующий отображаемому в текущий момент времени элементу данных в CarouselView, представляет собой темно-серый круг. Остальные элементы индикатора представляет собой светло-серые круги.

Для демонстрации возможностей мобильного программного приложения по работе со списками и коллекциями с помощью элементов управления CarouselView и IndicatorView создается проект CarouselViewDemo для мультиплатформенного приложения Xamarin.Forms с использованием шаблона «Пустой». После создания проекта сначала необходимо зайти в диалоговое окно «Обозреватель решений», выделить и открыть проект CarouselViewDemo, а затем с использованием рассмотренных ранее действий создать классы Monkey и MonkeysViewModel.

В диалоговом окне «Обозреватель решений» необходимо выделить и открыть файл Monkey.cs и набрать программный код Си Шарп, приведенный далее. В программном коде приведены свойства Name, Location, Details, ImageUrl класса Monkey, на которые будет производиться ссылаться из файлов MainPage.xaml и MonkeysViewModel.cs.

```
namespace CarouselViewDemo
{
   public class Monkey
   {
      public string Name {get; set;}
      public string Location {get; set;}
      public string Details {get; set;}
      public string ImageUrl {get; set;}
   }
}
```

В диалоговом окне «Обозреватель решений» необходимо выделить и открыть файл MonkeysViewModel.cs и набрать программный код Си Шарп, приведенный далее.

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Windows.Input;
using Xamarin.Forms;

namespace CarouselViewDemo
{
   public class MonkeysViewModel
   {
      private readonly IList<Monkey> source;
      public ObservableCollection<Monkey>
                              Monkeys { get; private set; }
      public MonkeysViewModel()
      {
         source = new List<Monkey>();
         CreateMonkeyCollection();
      }
      void CreateMonkeyCollection()
      {
         source.Add(new Monkey
         {   Name = "Капучин",
            Location = "Центральная и Южная Америка",
            Details = "Обезьяны капуцины - это обезьяны " +
            "Нового Света из подсемейства Cebinae",
           ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
            "commons/thumb/4/40/Capuchin_Costa_Rica.jpg/" +
            "200px-Capuchin_Costa_Rica.jpg"
```

```
        });
        source.Add(new Monkey
        {   Name = "Голубая обезьяна",
            Location = "Центральная и восточная Африка",
             Details =
         "Синяя обезьяна, или обезьяна в диадеме, - это вид" +
         " обезьян Старого Света, произрастающий в " +
         "Центральной и Восточной Африке, от верхнего " +
         "бассейна реки Конго на восток до " +
         "Восточно-Африканского разлома и на юге до " +
         "северной Анголы и Замбии",
        ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
"commons/thumb/8/83/BlueMonkey.jpg/220px-BlueMonkey.jpg"
        });
        source.Add(new Monkey
        {  Name = "Золотистый львиный тамарин",
            Location = "Бразилия",
          Details = "Известен также как золотая мартышка, " +
  "представляет собой небольшую обезьяну Нового Света " +
             "из семейства Callitrichidae",
           ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
   "commons/thumb/8/87/Golden_lion_tamarin_portrait3.jpg/" +
             "220px-Golden_lion_tamarin_portrait3.jpg"
        });
        source.Add(new Monkey
        {  Name = "Ревуны",
            Location = "Центральная и Южная Америка",
      Details = "Обезьяны-ревуны - одни из самых крупных " +
       "обезьян Нового Света. В настоящее время известно " +
       "пятнадцать видов. Ранее классифицированные в " +
             "семействе Cebidae, теперь они помещены в " +
             "семейство Atelidae",
           ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
             "commons/thumb/0/0d/Alouatta_guariba.jpg/" +
             "200px-Alouatta_guariba.jpg"
        });
        source.Add(new Monkey
        {  Name = "Японский макак",
            Location = "Япония",
            Details = "Японские макаки - наземные обезьяны " +
     "Старого Света, обитающие в Японии. " +
     "Их также иногда называют снежными обезьянами, " +
     "потому что они живут в районах, где снег покрывает " +
     "землю в течение нескольких месяцев каждый год",
       ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
```

```
        "commons/thumb/c/c1/Macaca_fuscata_fuscata1.jpg/" +
        "220px-Macaca_fuscata_fuscata1.jpg"
        });
        source.Add(new Monkey
        {  Name = "Носач",
           Location = "Борнео",
     Details = "Обезьяна-носач или длинноносая обезьяна, " +
     "известная как бекантан на малайском языке, - " +
    "это красновато-коричневая древесная обезьяна " +
    "Старого Света, которая является эндемиком острова " +
           "Борнео в юго-восточной Азии",
        ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
     "commons/thumb/e/e5/Proboscis_Monkey_in_Borneo.jpg/" +
           "250px-Proboscis_Monkey_in_Borneo.jpg"
        });
        source.Add(new Monkey
        {  Name = "Красноголовый доук",
           Location = "Вьетнам и Лаос",
           Details = "Красноголовый доук - это вид обезьян " +
           "Старого Света. Эту обезьяну иногда называют " +
"«костюмированной обезьяной». От колен до щиколоток " +
"он носит бордово-красные «чулки» и, кажется, " +
"носит белые перчатки до предплечья. " +
"Его наряд завершают черные руки и ноги. " +
"Золотое лицо обрамляет белый ерш, который у " +
"самцов значительно пушистее",
        ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
           "commons/thumb/9/9f/Portrait_of_a_Douc.jpg/" +
           "159px-Portrait_of_a_Douc.jpg"
        });
        source.Add(new Monkey
        {Name = "Черная курносая обезьяна",
          Location = "Китай",
    Details = "Черная курносая обезьяна, также известная " +
     "как курносая обезьяна Юньнани, является " +
     "вымирающим видом приматов семейства " +
     "Cercopithecidae. Это эндемик Китая, где он известен " +
     " местным жителям как обезьяна с золотыми " +
     " волосами Юньнани и обезьяна с черными " +
     " золотыми волосами. Ему угрожает потеря среды " +
     " обитания",
      ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
                "commons/thumb/5/59/RhinopitecusBieti.jpg/" +
                "320px-RhinopitecusBieti.jpg"
        });
```

```
            source.Add(new Monkey
            {  Name = "Гелада",
               Location = "Эфиопия",
               Details = "Гелада, которую иногда называют " +
                "кровоточащей обезьяной или бабуином гелада, - " +
              "это вид обезьян Старого Света, обитающий только " +
              "в Эфиопском нагорье, с большими популяциями в " +
              "горах Семьен. Как и его близкие родственники " +
              "павианы, он в основном наземный, проводя " +
              "большую часть времени в поисках пищи на лугах",
              ImageUrl = "https://upload.wikimedia.org/wikipedia/" +
                "commons/thumb/1/13/Gelada-Pavian.jpg/" +
                "320px-Gelada-Pavian.jpg"
              });
              Monkeys = new ObservableCollection<Monkey>(source);
        }
    }
}
```

В программном коде сначала объявляются свойство source, представляющее собой список из элементов типа Monkey, доступное только для чтения. Также объявлено свойство Monkeys, представляющее собой коллекцию типа ObservableCollection, состоящую из элементов типа Monkey. Производится создание визуальной коллекции элементов Monkeys, которая состоит из 9 элементов типа Monkey. Отметим, что в программном коде, приведенном в презентации значения свойств Details и ImageUrl типа String для удобства демонстрации приведены в виде многострочного текста. При наборе программного кода в среде разработки необходимо набирать текст в одну строку. В противном случае использованием свойств ImageUrl изображения коллекции Monkeys не будут загружаться, а отображение текста с помощью свойства Details будет отличаться от требуемого.

В диалоговом окне «Обозреватель решений» необходимо выделить и открыть файл MainPage.xaml и набрать программный код XAML, приведенный далее.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin.com/schemas/2014/forms/design"
          xmlns:mc="http://schemas.openxmlformats.org/markup-
                                                compatibility/2006"
          xmlns:viewmodels="clr-namespace:CarouselViewDemo"
          x:Class=" CarouselViewDemo.MainPage"
          BackgroundColor="Cyan">

    <ContentPage.BindingContext>
```

```
    <viewmodels:MonkeysViewModel />
</ContentPage.BindingContext>
<StackLayout Margin="10">
    <Label Text="Работа с CarouselView и IndicatorView"
           FontSize="30"
           TextColor="Blue"
           FontAttributes="Bold"
           HorizontalTextAlignment="Center"/>
    <CarouselView ItemsSource="{Binding Monkeys}"
                  IndicatorView="indicatorView"
                  EmptyView="Нет данных для отображения">
        <CarouselView.ItemTemplate>
            <DataTemplate>
                <StackLayout>
                    <Frame HasShadow="True"
                           BorderColor="Violet"
                           BackgroundColor="LightYellow"
                           CornerRadius="5"
                           Margin="50"
                           HeightRequest="500"
                           HorizontalOptions="Center"
                           VerticalOptions="CenterAndExpand">
                        <StackLayout>
                            <Label Text="{Binding Name}"
                                   TextColor="Red"
                                   FontAttributes="Bold"
                                   FontSize="Large"
                                   HorizontalOptions="Center"
                                   VerticalOptions="Center"
                                   HorizontalTextAlignment="Center"/>
                            <Image Source="{Binding ImageUrl}"
                                   Aspect="AspectFill"
                                   HeightRequest="150"
                                   WidthRequest="150"
                                   HorizontalOptions="Center" />
                            <Label Text="{Binding Location}"
                                   TextColor="Green"
                                   HorizontalOptions="Center"
                                   HorizontalTextAlignment="Center"/>
                            <Label Text="{Binding Details}"
                                   TextColor="Black"
                                   FontAttributes="Italic"
                                   HorizontalOptions="Center"
                                   HorizontalTextAlignment="Center"
                                   MaxLines="10"/>
```

```
                </StackLayout>
            </Frame>
        </StackLayout>
    </DataTemplate>
</CarouselView.ItemTemplate>
</CarouselView>
<IndicatorView x:Name="indicatorView"
               MaximumVisible="9"
               Margin="0,0,0,40"
               IndicatorColor="Red"
               SelectedIndicatorColor="Black"
               HorizontalOptions="Center" />
</StackLayout>
</ContentPage>
```

В программном коде определено, что для страницы типа ContentPage используется визуальная коллекция данных MonkeysViewModel, которая формируется с помощью программного кода, приведенного в файле MonkeysViewMode.cs.

Внутри страницы расположен макет типа StackLayout, содержащий три элемента управления (Label, CarouselView и IndicatorView).

Элемент управления Label предназначен для отображения названия задачи, которую решает программное приложение.

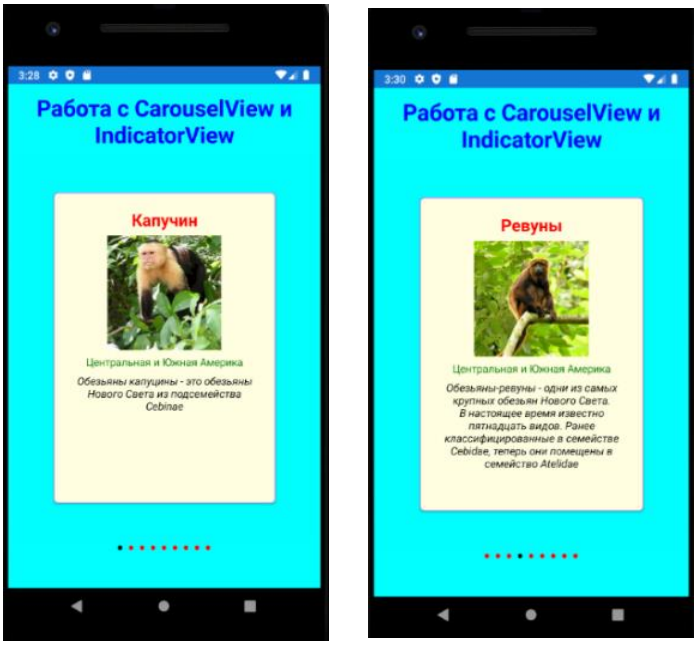
В элемент управления CarouselView с помощью привязки Binding загружается визуальная коллекция данных Monkeys. К элементу управления CarouselView привязывается элемент управления IndicatorView. Также для него определяется значение свойства EmptyView в виде строки на случай, если в коллекции Monkeys не будет элементов для отображения.

Внешний вид каждого элемента коллекции в элементе управления CarouselView задается путем присвоения свойству ItemTemplate шаблона данных DataTemplate. Шаблон данных DataTemplate содержит макет Frame, формирующий обрамление для находящегося внутри макета StackLayout. В макете StackLayout отображаются три элемента управления Label для вывода текста, соответствующего свойствам Name, Location и Details отображаемого в текущий момент времени элемента данных Monkey коллекции Monkeys. Также в макете StackLayout отображается элемент управления Image, в который загружается изображение, соответствующее свойству ImageUrl для отображаемого в текущий момент времени элемента данных Monkey. Свойству ImageUrl каждого элемента коллекции в соответствии с файлом MonkeysViewMode.cs соответствует ссылка на источник в Интернете.

Элемент управления IndicatorView отображается под элементом управления CarouselView. Элемент управления IndicatorView отображает 9 индикаторов в соответствии с количеством элементов в коллекции Monkeys и прикрепляется к элементу управления CarouselView с помощью свойства IndicatorView. Каждый индикатор представляет собой красный круг, а

индикатор, соответствующий отображаемому в текущий момент времени элементу коллекции, имеет черный цвет.

Для демонстрации возможностей мобильного программного приложения по работе со списками и коллекциями с использованием элементов управления CarouselView и IndicatorView сначала необходимо собрать решение, а затем, при отсутствии ошибок, запустить программное приложение. При запуске программного приложения отображается название решаемой задачи. Также после скачивания изображения из источника в Интернете отображается первый элемент коллекции Monkeys («Капучин»). Под отображаемым элементом коллекции находятся индикаторы, количество которых совпадает с числом элементов коллекции. При этом, первому элементу коллекции, отображаемому после запуска программного приложения, соответствует расположенный слева индикатор черного цвета. Остальные индикаторы красного цвета (рис. а).



а) б)

Рисунок. Демонстрация возможностей мобильного программного приложения по работе со списками и коллекциями с использованием элементов управления CarouselView и IndicatorView

Элементы коллекции можно прокручивать влево и вправо. В случае смены отображаемого элемента коллекции изменяет цвет на красный и соответствующий ему индикатор (рис. б). С помощью рассмотренного ранее шаблона DataTemplate элементы коллекции отображаются одинаковым образом на пользовательском интерфейсе.