



## DataTemplate и сложные объекты в ListView

Последнее обновление: 29.03.2017



Помогите им найти вашу  
компанию в Google!

НАЧНИТЕ СЕЙЧАС

Компенсируем  
до \$60

Google AdWords

В прошлой теме был рассмотрен вывод в ListView простейших данных - массива строк. На самом деле в простейшем варианте ListView получает для каждого объекта списка стоковое представление и выводит его в отдельной ячейке. Однако, как правило, данные, которыми манипулируют приложения, представляют более сложные по структуре объекты, которые могут содержать множество свойств. И для организации отображения каждого объекта в ListView определяется шаблон данных или DataTemplate.

Рассмотрим, как выполнять привязку к списку объектов на примере ListView. Для создания объектов определим класс Phone:

```
public class Phone
{
    public string Title { get; set; }
    public string Company { get; set; }
    public int Price { get; set; }
}
```

Теперь определим следующий класс страницы:

```
public partial class MainPage : ContentPage
{
    public List<Phone> Phones { get; set; }

    public MainPage()
    {
        Label header = new Label
        {
            Text = "Список моделей",
            FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            HorizontalOptions = LayoutOptions.Center
        };

        Phones = new List<Phone>
        {
            new Phone {Title="Galaxy S8", Company="Samsung", Price=48000 },
            new Phone {Title="Huawei P10", Company="Huawei", Price=35000 },
            new Phone {Title="HTC U Ultra", Company="HTC", Price=42000 },
            new Phone {Title="iPhone 7", Company="Apple", Price=52000 }
        };

        ListView listView = new ListView
        {
            HasUnevenRows = true,
            // Определяем источник данных
            ItemsSource = Phones,

            // Определяем формат отображения данных
            ItemTemplate = new DataTemplate(() =>
            {
                // привязка к свойству Name
                Label titleLabel = new Label { FontSize=18 };
                titleLabel.SetBinding(Label.TextProperty, "Title");

                // привязка к свойству Company
                Label companyLabel = new Label();
                companyLabel.SetBinding(Label.TextProperty, "Company");

                // привязка к свойству Price
                Label priceLabel = new Label();
            });
        }
    }
}
```

```

        priceLabel.SetBinding(Label.TextProperty, "Price");

        // создаем объект ViewCell.
        return new ViewCell
        {
            View = new StackLayout
            {
                Padding = new Thickness(0, 5),
                Orientation = StackOrientation.Vertical,
                Children = { titleLabel, companyLabel, priceLabel }
            }
        };
    })
};
this.Content = new StackLayout { Children = { header, listView } };
}
}

```

Для создания списков сложных объектов нам надо использовать три свойства. `ItemsSource` определяет, какие данные будет содержать список. Свойство `HasUnevenRows` со значением `true` позволит динамически устанавливать высоту строк в `ListView` в зависимости от размера содержимого. А свойство `ItemTemplate` определяет, как эти данные будут отображаться. `ItemTemplate` представляет объект **DataTemplate**, который в качестве параметра принимает делегат. Содержимое `DataTemplate` представляет одну из ячеек, то есть класс `Cell`. В данном случае это тип `ViewCell`, поэтому в качестве итога возвращается объект **ViewCell**, определяющий структуру строки в `ListView`.

Внутри `ViewCell` внутренние элементы упавляются контейнером `StackLayout`, а отдельные элементы `Label` внутри `StackLayout` привязаны к отдельным свойствам объекта `Phone`.



Определение `DataTemplate` в коде 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"
    xmlns:local="clr-namespace:HelloApp;assembly=HelloApp"
    x:Class="HelloApp.MainPage">
    <StackLayout>
        <Label Text="{Binding Source={x:Reference Name=phonesList}, Path=SelectedItem.Title}"
            FontSize="Large" />
        <ListView x:Name="phonesList"
            HasUnevenRows="True"
            ItemsSource="{Binding Phones}"
            ItemTapped="OnItemTapped">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout>

```

```

        <Label Text="{Binding Company}" FontSize="18" />
        <Label Text="{Binding Title}" />
        <Label Text="{Binding Price}" />
    </StackLayout>
</ViewCell.View>
</ViewCell>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
</StackLayout>
</ContentPage>

```

Здесь практически тоже самое, за исключением некоторых моментов. Так, атрибут `ItemsSource="{Binding Phones}"` устанавливает привязку к свойству `Phones`, которое мы далее определим.

И атрибут `ItemTapped="OnItemTapped"` устанавливает обработчик, который будет срабатывать при нажатии на один из элементов списка.

Для определения визуального шаблона выводимых данных применяется объект `DataTemplate`, в котором через элемент `ViewCell` устанавливается формат отображения - в виде вертикального стека элементов `Label`, каждый из которых привязан к одному из свойств объекта `Phone`:

```

<DataTemplate>
  <ViewCell>
    <ViewCell.View>
      <StackLayout>
        <Label Text="{Binding Company}" FontSize="18" />
        <Label Text="{Binding Title}" />
        <Label Text="{Binding Price}" />
      </StackLayout>
    </ViewCell.View>
  </ViewCell>
</DataTemplate>

```

Также надо отметить, что сверху определен элемент `Label`, текст которого привязан к названию выбранной в списке модели:

```

<Label Text="{Binding Source={x:Reference Name=phonesList}, Path=SelectedItem.Title}"
  FontSize="Large" />

```

Затем в коде на `C#` определим свойство `Phones`, к которому будет привязан список `ListView`, и обработчик нажатия на объект списка:

```

using System.Collections.Generic;
using Xamarin.Forms;

namespace HelloApp
{
    public partial class MainPage : ContentPage
    {
        public List<Phone> Phones { get; set; }
        public MainPage()
        {
            InitializeComponent();
            Phones = new List<Phone>
            {
                new Phone {Title="Galaxy S8", Company="Samsung", Price=48000 },
                new Phone {Title="Huawei P10", Company="Huawei", Price=35000 },
                new Phone {Title="HTC U Ultra", Company="HTC", Price=42000 },
                new Phone {Title="iPhone 7", Company="Apple", Price=52000 }
            };
            this.BindingContext = this;
        }

        public async void OnItemTapped(object sender, ItemTappedEventArgs e)
        {
            Phone selectedPhone = e.Item as Phone;
            if (selectedPhone != null)
                await DisplayAlert("Выбранная модель", $"{selectedPhone.Company} - {selectedPhone.Title}", "OK");
        }
    }
}

```

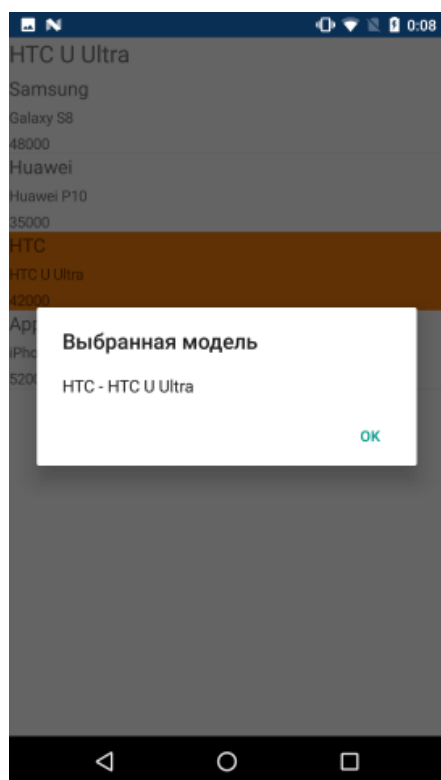
Установка в качестве контекста данных текущего объекта:

```
this.BindingContext = this;
```

Позволит использовать все свойства этого объекта (в данном случае свойство `Phones`) в коде XAML.

В обработчике события `OnItemTapped` получаем выбранный объект. И так как каждый объект в `DataTemplate` представляет тип `Phone`, то мы можем привести значение `e.Item` к типу `Phone`.

В итоге, когда мы запустим приложение, то в списке отобразится несколько начальных объектов, а название выбранного объекта появится в верхе страницы.



## Освойте вёрстку сейчас

Сверстайте ваш первый сайт под руководством опытного наставника. [htmlacademy.ru](http://htmlacademy.ru)



[Назад](#) [Содержание](#) [Вперед](#)

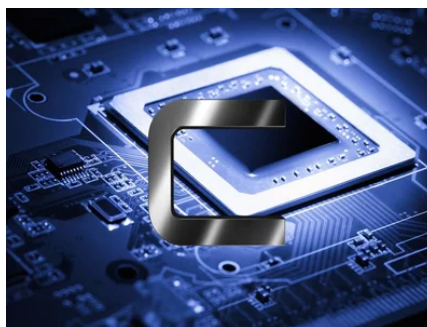


**ОБРАЗОВАТЕЛЬНЫЙ  
ЦЕНТР ПВТ**

**КУРС iOS  
РАЗРАБОТЧИКА**  
Высокая востребованность  
на рынке IT!

### Микроконтроллеры для начинающих

 [mcu-c.ru](http://mcu-c.ru)



Яндекс.Директ

11 Комментариев [metanit.com](http://metanit.com)

 Войти ▾

 Рекомендовать  Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя



**Vyacheslav Agibalov** • год назад

Как забиндить обычную строку?

^ | v • Ответить • Поделиться ›



**Metanit** Модератор → Vyacheslav Agibalov • год назад

определить строку как ресурс и установить в качестве ресурса для элемента управления

^ | v • Ответить • Поделиться ›



**Ярослав Орлов** • год назад

Добрый вечер, скажите какова максимальная вложенность привязки списков?

Например Response.Data.CurrentCondition как список читается

а если Response.Data.Weather.Astronomy то уже в ListView не отображаются данные. Скажите - это где-то я накосячил или существует предел вложенности?

^ | v • Ответить • Поделиться ›



**Metanit** Модератор → Ярослав Орлов • год назад

как-то с такой проблемой не сталкивался, поэтому ничего не могу сказать по поводу уровней вложенности

^ | v • Ответить • Поделиться ›



**Ярослав Орлов** → Metanit • год назад

Проблема решена, мой косяк, с вложенностью нет не каких ограничений (может быть сколько угодно), в данном случае проблема была просто из за многочисленной вложенности IEnumerable коллекций и просто недосмотрел к чему я обращаюсь (пытался вывести список в единичный элемент вместо к примеру ListView), из за сложности модели получилось так что у меня список списков был и надо было корректно это прибиндить. Спасибо за ваш превосходный ресурс!

^ | v • Ответить • Поделиться ›



**Baymen** • год назад

Хочу объединить с прошлым уроком и добавить в каждый итем кнопку обновить, и чтоб увеличивалась цена конкретной модели (в зависимости, где нажал кнопку) Как мне получить номер итема, где была нажата кнопка? Или можно по другому получить конкретный итем списка?

^ | v • Ответить • Поделиться ›



**Metanit** Модератор → Baymen • год назад

а вот же мы получаем конкретный элемент списка

Phone selectedPhone = e.Item as Phone;

далее мы можем с ним делать все, что угодно - изменять свойства и т.д. Только для изменения свойств он должен реализовать интерфейс INotifyPropertyChanged

^ | v • Ответить • Поделиться ›



**Baymen** → Metanit • год назад

Это при клике на элемент ListView, а если сделать кнопку, то в нём не будет возможности получить Item

^ | v • Ответить • Поделиться ›



**Metanit** Модератор → Baymen • год назад

тогда вам надо привязывать к кнопке параметр команды

далее в теме про MVVM есть примеры - <http://metanit.com/sharp/xa...>

^ | v • Ответить • Поделиться ›



**levivas** • год назад

Сделайте такой же пример только на C#

^ | v • Ответить • Поделиться ›

**Макс** → levivas • год назад


Это и есть C#

^ | v • Ответить • Поделиться ›


ТАКЖЕ НА METANIT.COM

**Руководство по языку Go**


2 комментария • месяц назад

 **Alexey** — О, спасибо большое, что уделили внимание этому языку**Паттерны в C# и .NET | Fluent Builder**





10 комментариев • 3 месяца назад

 **LamaK** — При том, что, если забыть какой-то параметр, что-то может обвалиться. По-хорошему, в метод Build() можно добавить проверку всех необходимых ...**React | React-router и webpack**

8 комментариев • 3 месяца назад

 **Юрий Демин** — да, установлен. решил проблему обновив S, но конкретной причины и связи с обновлением системы так и не нашел!) Спасибо за помощь и за ...**Vue.js | Props**

2 комментария • 4 месяца назад

 **Metanit** — нет, и так должно работать, проверил в Google Chrome, MS Edge, Firefox Подписаться  Добавить Disqus на свой сайт  Добавить Disqus  Конфиденциальность[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Copyright © metanit.com, 2012-2017. Все права защищены.