



Триггеры данных

Последнее обновление: 16.08.2016



	40,85 руб.	59,98 руб.	59,89 руб.	51,30 руб.	77,10 руб.	50,80 руб.	84 руб.

Триггеры данных (data triggers) позволяют отслеживать изменение свойств объектов. В Xamarin Forms триггеры данных представлены классом **DataTrigger**. Рассмотрим, как его использовать.

В качестве объекта, за свойствами которого будет следить триггер данных, возьмем следующий класс Phone:

```
using System.ComponentModel;

public class Phone : INotifyPropertyChanged
{
    private string title;
    private string company;
    private int price;

    public string Title
    {
        get { return title; }
        set
        {
            title = value;
            OnPropertyChanged("Title");
        }
    }
    public string Company
    {
        get { return company; }
        set
        {
            company = value;
            OnPropertyChanged("Company");
        }
    }
    public int Price
    {
        get { return price; }
        set
        {
            price = value;
            OnPropertyChanged("Price");
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged(string prop = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(prop));
    }
}
```

Важно отметить, что если мы хотим динамически отслеживать изменение свойств объекта, то класс, представляющий этот объект, должен реализовать интерфейс INotifyPropertyChanged.

В коде xaml у главной страницы определим пользовательский интерфейс с привязкой к объекту Phone:

```
<?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">
<ContentPage.Resources>
  <ResourceDictionary>
    <local:Phone x:Key="phone" Title="iPhone 7" Company="Apple" Price="56000" />
  </ResourceDictionary>
</ContentPage.Resources>
<Grid BindingContext="{StaticResource phone}">
  <Grid.RowDefinitions>
    <RowDefinition Height="30" />
    <RowDefinition Height="60" />
    <RowDefinition Height="30" />
    <RowDefinition Height="60" />
    <RowDefinition Height="30" />
    <RowDefinition Height="60" />
  </Grid.RowDefinitions>
  <Label Grid.Row="0" Text="Модель" />
  <Label Grid.Row="2" Text="Компания" />
  <Label Grid.Row="4" Text="Цена" />

  <Entry Grid.Row="1" Text="{Binding Path=Title}" />
  <Entry Grid.Row="3" Text="{Binding Path=Company}">
    <Entry.Triggers>
      <DataTrigger Binding="{Binding Company}" TargetType="Entry" Value="Microsoft">

        <Setter Property="TextColor" Value="#004D40" />
        <Setter Property="BackgroundColor" Value="#80CBC4" />

      </DataTrigger>
    </Entry.Triggers>
  </Entry>
  <Entry Grid.Row="5" Text="{Binding Path=Price}" />
</Grid>
</ContentPage>

```

Здесь установлена привязка трех элементов Entry к разным свойствам объекта Phone, который определен в ресурсах. Причем для второго элемента Entry добавлен триггер данных:

```

<Entry Grid.Row="3" Text="{Binding Path=Company}">
  <Entry.Triggers>
    <DataTrigger Binding="{Binding Company}" TargetType="Entry" Value="Microsoft">

      <Setter Property="TextColor" Value="#004D40" />
      <Setter Property="BackgroundColor" Value="#80CBC4" />

    </DataTrigger>
  </Entry.Triggers>
</Entry>

```

У триггера данных в обязательном порядке должно быть установлено свойство TargetType, которое указывает на тип элемента, к которому применяется триггер.

Также у триггера через свойство Binding задается привязка к определенному свойству объекта Phone (в данном случае свойство Company). А через свойство Value определяется значение, при котором будет срабатывать триггер. То есть в данном случае триггер будет срабатывать, когда свойство Company объекта Phone получит значение "Microsoft".

Вся работа триггера заключается в установке свойств объекта Entry. Для этого у триггера в коллекцию Setters добавляются объекты Setter, которые указывают, какое свойство будет изменяться и какое значение оно получит. То есть в данном случае, когда свойство Company объекта Phone получит значение "Microsoft", свойство BackgroundColor у Entry получит значение "#80CBC4", свойство TextColor - значение "#004D40".

В данном случае триггер привязан к тому же свойству, к которому привязан и объект Entry. Однако это необязательно, они могут быть привязаны к разным свойствам. И теперь, если мы запустим приложения и введем в поле, которое представляет компанию, строку "Microsoft", то сработает наш триггер данных:



Создание аналогичного триггера данных в коде c#:

```
Entry companyValueEntry = new Entry();
Binding companyBinding = new Binding { Source = phone, Path = "Company" };
companyValueEntry.SetBinding(Entry.TextProperty, companyBinding);

DataTrigger dataTrigger = new DataTrigger(typeof(Entry))
{
    Binding = new Binding { Source = phone, Path = "Company" },
    Value = "Microsoft"
};
dataTrigger.Setters.Add(new Setter { Property = Entry.BackgroundColorProperty, Value = Color.FromHex("#80CBC4") });
dataTrigger.Setters.Add(new Setter { Property = Entry.TextColorProperty, Value = Color.FromHex("#004D40") });

companyValueEntry.Triggers.Add(dataTrigger);
```

Полный пример:

```
using Xamarin.Forms;

namespace HelloApp
{
    public partial class MainPage : ContentPage
    {
        Phone phone;
        public MainPage()
        {
            phone = new Phone { Title = "iPhone 7", Company = "Apple", Price = 56000 };

            Grid grid = new Grid
            {
                RowDefinitions =
                {
                    new RowDefinition { Height = 50 },
                    new RowDefinition { Height = 60 },
                    new RowDefinition { Height = 50 },
                    new RowDefinition { Height = 60 },
                    new RowDefinition { Height = 50 },
                    new RowDefinition { Height = 60 }
                }
            };

            Label titleHeaderLabel = new Label { Text = "Модель" };
            Label companyHeaderLabel = new Label { Text = "Компания" };
            Label priceHeaderLabel = new Label { Text = "Цена" };

            Entry titleValueEntry = new Entry();
            Binding titleBinding = new Binding { Source = phone, Path = "Title" };
            titleValueEntry.SetBinding(Entry.TextProperty, titleBinding);
```

```

Entry companyValueEntry = new Entry();
Binding companyBinding = new Binding { Source = phone, Path = "Company" };
companyValueEntry.SetBinding(Entry.TextProperty, companyBinding);

DataTrigger dataTrigger = new DataTrigger(typeof(Entry))
{
    Binding = new Binding { Source = phone, Path = "Company" },
    Value = "Microsoft"
};
dataTrigger.Setters.Add(new Setter { Property = Entry.BackgroundColorProperty, Value =
Color.FromHex("#80CBC4") });
dataTrigger.Setters.Add(new Setter { Property = Entry.TextColorProperty, Value = Color.FromHex("#004D40")
});

companyValueEntry.Triggers.Add(dataTrigger);

Entry priceValueEntry = new Entry();
Binding priceBinding = new Binding { Source = phone, Path = "Price" };
priceValueEntry.SetBinding(Entry.TextProperty, priceBinding);

grid.Children.Add(titleHeaderLabel, 0,0);
grid.Children.Add(companyHeaderLabel, 0, 2);
grid.Children.Add(priceHeaderLabel, 0, 4);

grid.Children.Add(titleValueEntry, 0, 1);
grid.Children.Add(companyValueEntry, 0, 3);
grid.Children.Add(priceValueEntry, 0, 5);

Content = grid;
    }
}
}

```

Таким образом, триггеры данных позволяют внести некоторое дополнительное форматирование в зависимости от значения свойства объекта. Особенно полезной такая функция может оказаться при отображении списков объектов. Допустим, в коде страницы определено свойство Phones, которое содержит некоторый набор объектов:

```

using System.Collections.ObjectModel;
using Xamarin.Forms;

namespace HelloApp
{
    public partial class MainPage : ContentPage
    {
        public ObservableCollection<Phone> Phones { get; set; }
        public MainPage()
        {
            InitializeComponent();

            Phones = new ObservableCollection<Phone>
            {
                new Phone { Title = "HTC U Ultra", Company = "HTC", Price = 36000 },
                new Phone {Title="Huawei P10", Company="Huawei", Price=35000 },
                new Phone {Title="iPhone 6S", Company="Apple", Price=42000 },
                new Phone {Title="LG G 6", Company="LG", Price=42000 },
                new Phone {Title="iPhone 7", Company="Apple", Price=52000 }
            };
            this.BindingContext = this;
        }
    }
}

```

В коде XAML у страницы применим триггер данных, чтобы выделить цветом те объекты, которые принадлежат определенной компании:

```

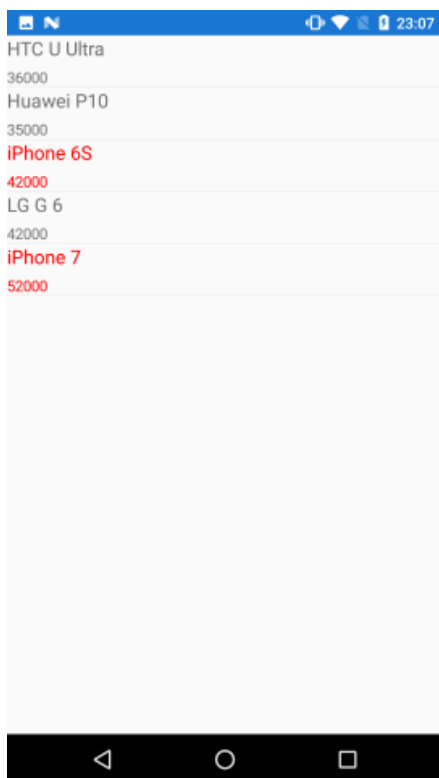
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:HelloApp;assembly=HelloApp"
    x:Class="HelloApp.MainPage">
    <StackLayout>
        <ListView HasUnevenRows="True" ItemsSource="{Binding Phones}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout>
                            <Label Text="{Binding Title}" FontSize="Medium">

```

```

<Label.Triggers>
  <DataTrigger TargetType="Label"
    Binding="{Binding Company}" Value="Apple">
    <Setter Property="TextColor" Value="Red" />
  </DataTrigger>
</Label.Triggers>
</Label>
<Label Text="{Binding Price}">
  <Label.Triggers>
    <DataTrigger TargetType="Label"
      Binding="{Binding Company}" Value="Apple">
      <Setter Property="TextColor" Value="Red" />
    </DataTrigger>
  </Label.Triggers>
</Label>
</StackLayout>
</ViewCell>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
</StackLayout>
</ContentPage>

```



Яндекс.Директ

	40,85 руб.	88 руб.	59,89 руб.	70 руб.	59,98 руб.	51,30 руб.	57,30 руб. ООО "Триовист"
--	------------	---------	------------	---------	------------	------------	------------------------------

0 Комментариев metanit.com

1 Войти ▾

♥ Рекомендовать  Поделиться

Лучшее в начале ▾



Начать обсуждение...

ВОЙТИ С ПОМОЩЬЮ

или ЧЕРЕЗ DISQUS 

Имя

Прокомментируйте первым.

ТАКЖЕ НА METANIT.COM

Go | Соответствие интерфейсу

1 комментарий • месяц назад



Пу — Еще давай. Не останавливайся, пожалуйста. Это гениальный язык. Он, конечно, не божественен, как шарп, но гениальный.

C# и .NET | Раннее и позднее связывание

4 комментариев • 2 месяца назад



dev loop — спасибо, ответили и на мой вопрос тоже)

Паттерны в C# и .NET | Fluent Builder

10 комментариев • 3 месяца назад






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

Kotlin | Руководство

3 комментариев • 2 месяца назад



Артур Голояд — Просто шикарно, может подкинете материалов по работе с сетями на Kotlin?

 Подписаться  Добавь Disqus на свой сайтДобавить DisqusДобавить  Конфиденциальность