Привязка в Xamarin

Введение в привязку

Последнее обновление: 16.06.2016

Привязка данных (data binding) является одним из ключевых моментов платформы Xamarin Forms.

Привязка данных состоит из двух компонентов: источника (source) и цели (target). Привязка осуществляется от свойства источника к свойству цели. И когда происходит изменение источника, механизм привязки автоматически обновляет также и цель.

Цель привязки должна представлять объект **BindableObject**, а свойство, к которому осуществляется привязка, должно быть свойством **BindableProperty**. Поскольку большинство визуальных элементов в Xamarin Forms наследуются от класса BindableObject, то в качестве цели привязки будут, как правило, выступать визуальные элементы.

А вот источником привязки может выступать любой объект языка C#. Однако, надо понимать, что цель привязки должна автоматически изменяться при изменении источника, поэтому нам нужно извещать систему о изменении свойств источника привязки. В Xamarin, да и вообще на платформе .NET, в качестве подобного механизма извещения выступает интерфейс **INotifyPropertyChanged**. То есть нужно реализовать данный интерфейс в объекте-источнике.

Объект BindableObject как раз реализует `INotifyPropertyChanged`. Поэтому если источником привязки является стандартный визуальный элемент из Xamarin Forms, то автоматически будет изменяться и цель привязки. Ну если в качестве источника выступает не BindableObject, а какой-нибудь объект простого класса C#, то, как писалось выше, этот класс должен реализовать INotifyPropertyChanged.

Для установки привязки у объекта цели устанавливается свойство **BindingContext**. В качестве значения оно принимает источник привязки.

Рассмотрим на примерах, как устанавливается привязка в коде C# и в коде XAML.

Привязка в коде C#

Для привязки в коде после установки у цели привязки свойства **BindingContext** необходимо также у цели привязки вызвать метод **SetBinding()**, который свяжет свойство цели привязки со свойством источника.

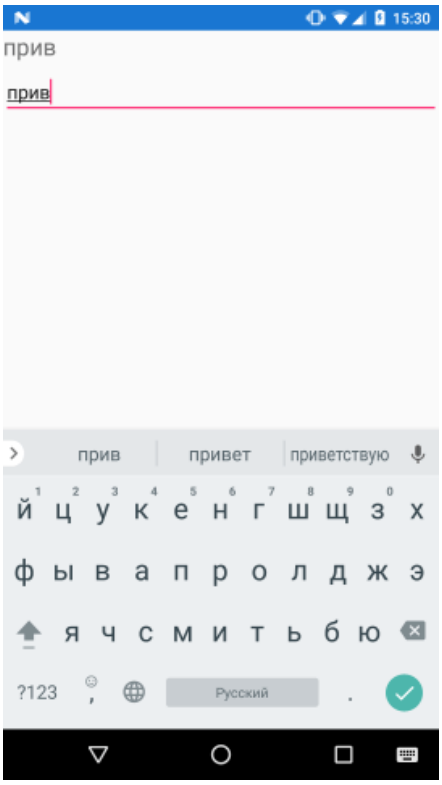
```
public partial class MainPage : ContentPage
{
    public MainPage()
    {
        Label label = new Label
        {
            FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label))
        };
        Entry entry = new Entry();

        // Устанавливаем привязку
        // источник привязки - entry, цель привязки - label
        label.BindingContext = entry;
        // Связываем свойства источника и цели
        label.SetBinding(Label.TextProperty, "Text");

        StackLayout stackLayout = new StackLayout()
        {
            Children = { label, entry}
        };
```

```
        Content = stackLayout;
    }
}
```

Выражение `label.SetBinding(Label.TextProperty, "Text")` устанавливает привязку свойства TextProperty у цели - элемента label к свойству `Text`.

Причем, что важно у объекта цели label привязка устанавливается именно у свойства `BindableProperty`, которым является `TextProperty`, а не просто для свойства `Text`.

В итоге при вводе данных в текстовое поле будет изменяться значение свойства Text у объекта entry. А это изменение автоматически скажется на объекте label и изменит его свойство TextProperty.



Привязка в XAML

Аналогично можно установить привязку и в xaml-коде.

```
<?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="HelloApp.MainPage">
  <StackLayout>
    <Label x:Name="label"
           BindingContext="{x:Reference Name=entryBox}"
           Text="{Binding Path=Text}"
           FontSize="Large" />
    <Entry x:Name="entryBox" />
  </StackLayout>
</ContentPage>
```

В XAML действует тот же принцип. Для установки привязки для объекта цели задается свойство **BindingContext**. Но теперь его значение имеет другую форму: `{x:Reference Name=entryBox}`. После расширения разметки `x:Reference` идет свойство `Name` и его значение - имя элемента источника. То есть в данном случае источником выступает объект с именем entryBox.

Далее устанавливается сама привязка к свойству:

```
Text="{Binding Path=Text}"
```

Здесь также устанавливается привязка к свойству Text. Выражение привязки заключается в фигурные скобки и состоит из слова **Binding**, после которого идет свойство `Path` и его значение - свойство объекта источника. То есть свойство Text объекта label привязано к свойству Text объекта entryBox.

Назад Содержание Вперед