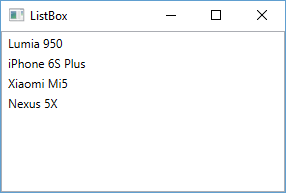
## Элементы управления списками

Эти элементы представлены в WPF довольно широко. Все они являются производными от класса **ItemsControl**, который в свою очередь является наследником класса Control. Все они содержат коллекцию элементов. Элементы могут быть напрямую добавлены в коллекцию, возможна также привязка некоторого массива данных к коллекции.

Возьмем простейший элемент-список - ListBox:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <Window x:Class="ControlsApp.MainWindow"          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"          xmlns:local="clr-namespace:ControlsApp"          xmlns:sys="clr-namespace:System;assembly=mscorlib"          mc:Ignorable="d"          Title="ListBox" Height="200" Width="300">      <Grid>          <ListBox Name="list">              <sys:String>Lumia 950</sys:String>              <sys:String>iPhone 6S Plus</sys:String>              <sys:String>Xiaomi Mi5</sys:String>              <sys:String>Nexus 5X</sys:String>          </ListBox>      </Grid>  </Window> |



Все элементы, размещенные внутри спискового элемента ListBox, представляют элементы списка.

Коллекция объектов внутри элемента-списка доступна в виде свойства **Items**. Для управления элементами из этой коллекции мы можем использовать следующие методы:

* **Add(object item)**: добавление элемента
* **Clear()**: полная очистка коллекции
* **Insert(int index, object item)**: вставка элемента по определенному индексу в коллекции
* **Remove(object item)**: удаление элемента
* **RemoveAt(int index)**: удаление элемента по индексу

А свойство **Count** позволяет узнать, сколько элементов в коллекции.

Например, применительно к вышеопределенному списку мы бы могли написать в коде C#:

|  |  |
| --- | --- |
| 1  2 | list.Items.Add("LG G5");  list.Items.RemoveAt(1); // удаляем второй элемент |

Нам необязательно вручную заполнять значения элемента управления списком, так как мы можем установить свойство **ItemsSource**, задав в качестве параметра коллекцию, из которой будет формироваться элемент управления списком. Например, в коде xaml-разметки определим пустой список:

|  |  |
| --- | --- |
| 1  2  3 | <Grid>      <ListBox Name="list" />  </Grid> |

А в файле отделенного кода выполним наполнение списка:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public partial class MainWindow : Window  {      public MainWindow()      {          InitializeComponent();            string[] phones = { "iPhone 6S", "Lumia 950", "Nexus 5X", "LG G4", "Xiaomi MI5", "HTC A9" };          list.ItemsSource = phones;      }  } |

Свойство ItemsSource в качестве значения принимает массив, хотя это моет быть и список типа List. И каждый элемент этого массива переходит в ListBox.

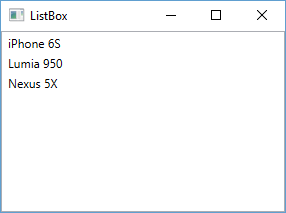
Еще одно важное свойство списковых элементов - это свойство **DisplayMemberPath**. Оно позволяет выбирать для отображения элементов значение одного из свойств объекта. Например, создадим в коде новый класс Phone:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | class Phone  {      public string Title { get; set; }      public string Company { get; set; }      public int Price { get; set; }  } |

Теперь создадим в xaml набор объектов этого класса Phone и выведем в списке значение свойства Title этих объектов:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <Window x:Class="ControlsApp.MainWindow"          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"          xmlns:local="clr-namespace:ControlsApp"          mc:Ignorable="d"          Title="ListBox" Height="220" Width="300">      <Grid Background="Lavender">          <ListBox Name="list" DisplayMemberPath="Title">              <local:Phone Title="iPhone 6S" Company="Apple" Price="54990" />              <local:Phone Title="Lumia 950" Company="Microsoft" Price="39990" />              <local:Phone Title="Nexus 5X" Company="Google" Price="29990" />          </ListBox>      </Grid>  </Window> |

Поскольку мы используем класс, определенный в текущем проекте, то соответственно у нас обязательно должно быть подключено пространство имен проекте: xmlns:local="clr-namespace:ControlsApp". В принципе по умолчанию WPF уже его подключает. Кроме того, чтобы не возникало проблем с разметкой XAML, желательно сделать перестроение проекта. И в итоге окно нам выведет названия смартфонов:



То же самое мы бы могли сделать программным способом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | list.ItemsSource = new List<Phone>  {      new Phone { Title="iPhone 6S", Company="Apple", Price=54990 },      new Phone {Title="Lumia 950", Company="Microsoft", Price=39990 },      new Phone {Title="Nexus 5X", Company="Google", Price=29990 }  };  list.DisplayMemberPath = "Title"; |

Все элементы управления списками поддерживают выделение входящих элементов. Выделенный элемент(ы) можно получить с помощью свойств **SelectedItem**(SelectedItems), а получить индекс выделенного элемента - с помощью свойства **SelectedIndex**. Свойство **SelectedValue** позволяет получить значение выделенного элемента.

При выделении элемента в списке генерируется событие **SelectionChanged**, которое мы можем обработать. Например, возьмем предыдущий список:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <ListBox Name="list" DisplayMemberPath="Title" SelectionChanged="list\_Selected">      <local:Phone Title="iPhone 6S" Company="Apple" Price="54990" />      <local:Phone Title="Lumia 950" Company="Microsoft" Price="39990" />      <local:Phone Title="Nexus 5X" Company="Google" Price="29990" />  </ListBox> |

А в файле кода определим обработчик для этого события:

|  |  |
| --- | --- |
| 1  2  3  4  5 | private void list\_Selected(object sender, RoutedEventArgs e)  {      Phone p = (Phone)list.SelectedItem;      MessageBox.Show(p.Title);  } |

Важно учитывать, что так как в разметке xaml в списке определены элементы Phone, то в коде мы можем привести объект list.SelectedItem к типу Phone.