Разработка приложений на платформе .NET

Лекция 15

Элементы управления в **WPF**

Элементы пользовательского интерфейса

- Элементы управления компоновкой. Контейнеры. **Panels**
- Элементы управления содержимым. **ContentControls**
- Элементы управления списком. **ItemsControls**
- Специализированные элементы управления

Компоновка

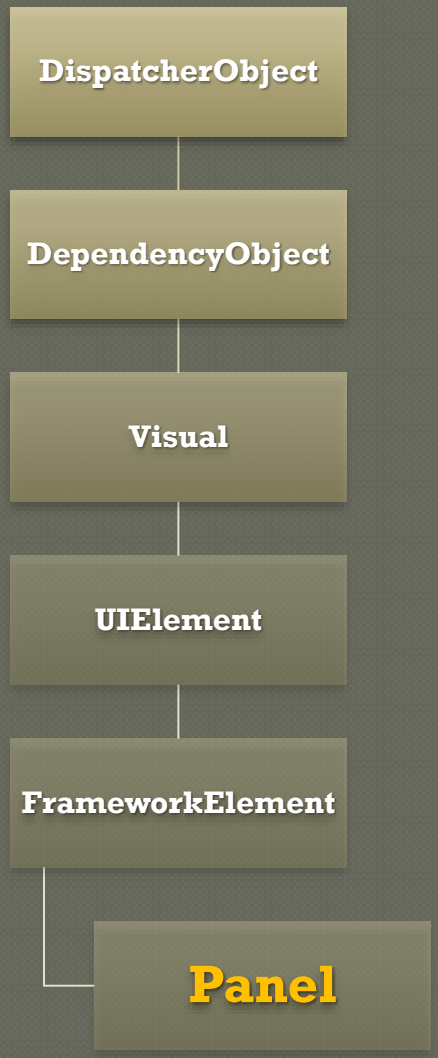
Элементы управления компоновкой (контейнеры)

Компоновка

- Рекомендации
  - Элементы не должны иметь явно установленных размеров
  - Элементы не должны указывать свою позицию в экранных координатах
  - Контейнеры компоновки распределяют доступное им пространство для размещения дочерних элементов
  - Контейнеры компоновки могут быть вложенными, и сами содержать другие контейнеры компоновки
- Отрисовка в два прохода
  - Измерение
  - Расположение

Базовый класс – Panel

- Все контейнеры компоновки – наследники от класса **System.Windows.Controls.Panel**
- Свойства **Panel**:
  - **Children** – коллекция дочерних элементов, располагающихся в контейнере
  - **Background** – фон
  - **IsItemsHost** – **true**, если контейнер используется для отображения элементов в **ItemsControl**



Контейнеры компоновки

Предназначены для расположения своих дочерних элементов

- **StackPanel** – размещение в горизонтальном или вертикальном стеке
- **WrapPanel** – размещение в строку или столбец с переносом (как текст)
- **DockPanel** – выстраивание элементов по краю контейнера
- **Grid** – расположение элементов в ячейках невидимой таблицы
- **UniformGrid** – расположение элементов в ячейках невидимой таблицы, но все столбцы и строки имеют одинаковые размеры
- **Canvas** – абсолютное покоординатное расположение дочерних элементов

Специальные

- **TabPanel** – вкладки в **TabControl**
- **ToolbarPanel** – группа кнопок в **Toolbar**
- **ToolbarOverflowPanel** – команды в раскрывающемся меню **Toolbar**
- **VirtualizingStackPanel** – **StackPanel** с оптимизацией накладных расходов при большой коллекции дочерних элементов
- **IncCanvas** – **Canvas** с поддержкой перьевого ввода

StackPanel

- Располагает свои дочерние элементы в одну строку или в одну колонку
- Сама панель по умолчанию занимает все свободное пространство

- Свойства:
  - Orientation – задает способ расположения дочерних элементов
    - Orientation="Horizontal" – горизонтальное
    - Orientation="Vertical" – вертикальное (по умолчанию)

- Пример:

```
<StackPanel>
    <Button>Save</Button>
    <Button>Cancel</Button>
</StackPanel>
```

WrapPanel

- Располагает элементы управления в доступном пространстве по строчкам (или по столбцам) с переносом не уместившихся элементов на новую строку (как текст)

- Свойства:
  - Orientation – задает способ расположения дочерних элементов
    - Orientation="Horizontal" – построчное расположение элементов (по умолчанию)
    - Orientation="Vertical" – расположение элементов по колонкам

- Пример:

```
<WrapPanel>
    <Image MaxHeight="100" Margin="5" Source="01.jpg" />
    <Image MaxHeight="100" Margin="5" Source="02.jpg" />
    <Image MaxHeight="100" Margin="5" Source="03.jpg" />
</WrapPanel>
```

DockPanel

- Пристыковывает элементы к одной из границ контейнера
- Важен порядок следования дочерних элементов

- Свойства:
  - LastChildFill – true, означает, что последний элемент заполняет все оставшееся пространство

- Присоединенные свойства, которые получают дочерние элементы:
  - DockPanel.Dock – указывает к какому краю контейнера необходимо пристыковать данный элемент управления
    - Возможные значения: Left, Right, Top, Bottom

- Пример:

```
<DockPanel LastChildFill="True">
    <ToolBar DockPanel.Dock="Top">    ...    </ToolBar>
    <Button DockPanel.Dock="Bottom" Content="Exit"/>
    <TextBox/>
</DockPanel>
```

Grid

- Располагает элементы в колонках и строках невидимой сетки
- Свойства:
  - ColumnDefenitions – содержит коллекцию определения колонок ColumnDefenition
  - RowDefenitions – содержит коллекцию определения строк RowDefenition
- Присоединенные свойства, которые получают дочерние элементы:
  - Grid.Row и Grid.Column – указывают контейнеру в какой строке и колонке необходимо расположить данный элемент управления.
    - Нумерация с 0. По умолчанию – 0
  - Grid.RowSpan и Grid. ColumnSpan – указывают контейнеру в скольких строках и колонках необходимо расположить данный элемент управления. Т.е. указывают, что элемент необходимо расположить в нескольких строках или колонках

Grid

Пример:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        <RowDefinition Height="2*"/>
    </Grid.RowDefinitions>
    <Button Grid.Column="0" Grid.Row="0">but 1</Button>
    <Button Grid.Column="1" Grid.Row="0">but 2</Button>
    <Button Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2">but
3</Button>
</Grid>
```

Задание Размера строк и колонок

- Свойства **RowDefinition**:
  - **Width** – ширина колонки
- Свойства **ColumnDefinition**:
  - **Height** – высота колонки
- Варианты задания высоты и ширины строк и колонок. Возможные значения:
  - Абсолютные размеры
    - `<ColumnDefinition Width="100"/>`
  - Автоматические размеры. Ширина или высота задаются в точности такими, какие необходимы для размещения дочерних элементов)
    - `<ColumnDefinition Width="Auto"/>`
  - Пропорциональные размеры. Пространство распределяется между группой столбцов или строк, которые заполняют все оставшееся пространство.
    - `<ColumnDefinition Width="*"/>`
    - Можно задать неравное пропорциональное изменение размеров
      - `<ColumnDefinition Width="1*"/>`
      - `<ColumnDefinition Width="2*"/>`

Grid

- Свойство **Grid UseLayoutRounding="True"** - заставляет размещать содержимое четко по ближайшим границам точек экрана, исключая размытие

- Элемент **GridSplitter** – представляет из себя раздельную полосу для **Grid**
  - Изменяет размер всей строки / колонки
  - Обычно располагают в отдельной колонке / строке и растягивают на всю строку / колонку

- Пример:

```
<GridSplitter Grid.Column="1" Grid.Row="0" Grid.RowSpan="2"
              Width="3"
              HorizontalAlignment="Center" VerticalAlignment="Stretch"/>
```

UniformGrid

- Располагает элементы внутри невидимой сетки из колонок и строк одинаковых размеров
- Свойства:
  - **Columns** – кол-во колонок, на которое необходимо разбить все пространство
  - **Rows** – кол-во строчек, на которое необходимо разбить все пространство
- Расположение элементов задается порядком их определения. Ячейки невидимой сетки заполняются слева направо и сверху вниз.
- Нет присоединенных свойств.

```
<UniformGrid Rows="2" Columns="2">
        <Image Margin="5" Source="01.jpg" />
        <Image Margin="5" Source="02.jpg" />
        <Image Margin="5" Source="03.jpg" />
        <Image Margin="5" Source="04.jpg" />
</UniformGrid>
```

Canvas

- Располагает элементы используя точные координаты
- Предоставляет дочерним элементам присоединенные свойства:
  - Canvas.Left, Canvas.Right, Canvas.Top, Canvas.Bottom – устанавливают точные положения элемента в координатах
  - Одновременно использовать Canvas.Left и Canvas.Right или Canvas.Top и Canvas.Bottom нельзя.
  - Элементы могут накладываться друг на друга
  - Panel.ZIndex – задает значение по оси Z. Элемент с большим значением Z будет виден при наложении элементов управления (для любого контейнера, а не тольrо для Canvas)
- Часто с Canvas для указания размеров дочернего элемента используют Width и Height. В противном случае элемент получает желаемые размеры.

```
<Canvas>
    <Button Canvas.Left="10" Canvas.Top="10" Content="Button 1"/>
    <Button Canvas.Left="10" Canvas.Bottom="20" Content="Button 2"/>
    <Button Canvas.Right="10" Canvas.Bottom="20" Content="Button 3"/>
</Canvas>
```

Специальные контейнеры

- TabPanel – вкладки в TabControl.
- ToolBarPanel – группа кнопок в Toolbar
- ToolBarOverflowPanel – команды в раскрывающемся меню Toolbar
- VirtualizingStackPanel – StackPanel с оптимизацией накладных расходов при большой коллекции дочерних элементов
- IncCanvas – Canvas с поддержкой перьевого ввода

Свойства, влияющие на компоновку

Свойства наследуются от FrameworkElement

- HorizontalAlignment, VerticalAlignment
  - Выравнивание элемента в выделенной области, если остается свободное место
  - Возможные значения: Stretch, Left, Right, Center
- Width, Height – ширина и высота
  - В аппаратно-независимых единицах
  - Обычно не задается, позволяя элементу занимать необходимое ему пространство
  - Height="350" Width="525”
- MinWidth, MinHeight, MaxWidth, MaxHeight – задает минимальные и максимальные размеры элемента
- Margin
  - Минимальное расстояние до соседей или до границ контейнера
  - Margin="2” – задает поля, одинаковые со всех сторон
  - Margin="2,5" – задает поля слева/справа (2) и сверху/снизу (5)
  - Margin="2,5,0,8" – задает поля слева, сверху, справа, снизу

Нет свойств, задающих координаты на экране

Демонстрация

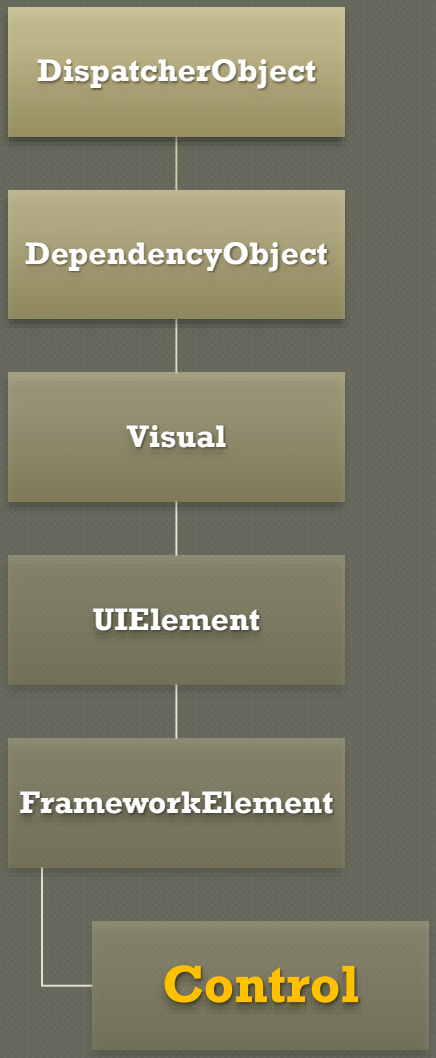
Работа с контейнерами

Content Controls

Элементы управления содержимым

Control

- Базовый класс для большинства элементов управления
- Свойства:
  - Background, Foreground – кисти фона и переднего плана (например, текста)
  - BorderBrush, BorderThickness – позволяют задать рамку вокруг элемента управления
  - FontFamily, FontSize, FontStyle, FontWeight, FontStretch – задают параметры текста (шрифт, размер, наклон, жирность, разреженность). Свойства наследуются дочерними элементами.



Элементы управления содержимым

- Наследники класса **ContentControl : Control**
- Могут содержать вложенный элемент и только один
- Свойство **Content** содержит вложенный элемент
  - **XAML:** <Button>Строка содержимого</Button>
  - **XAML:** <Button Content="Строка содержимого"></Button>
  - **C#:** button1.Content = "Строка содержимого";
- Свойство **Content** типа **object**
- Если в **Content** наследник от типа **UIElement**, то он отображается напрямую. Иначе для отображения вызывается **ToString**().

- Примеры:

```
<Button>
    <Image Source="d:\Yura\Камин.jpg"/>
</Button>
```

- Если нужно вложить несколько элементов нужно использовать контейнеры

```
<Button>
        <StackPanel Orientation="Vertical">
                <Image Source="d:\Yura\Камин.jpg"/>
                <TextBlock>Красивый камин</TextBlock>
        </StackPanel>
</Button>
```

Элементы управления содержимым

- **Label** - метка
- **ButtonBase** базовый класс для всех кнопок (имеет событие **Click**)
  - **Button** – кнопка
  - **ToggleButton** – залипающая кнопка (добавляет свойство **IsChecked**)
    - **CheckBox**
    - **RadioButton**
- **ToolTip** – всплывающее окно подсказки
- **SrollViewer** – добавляет прокрутку дочернему элементу
- **UserControl** – пользовательский элемент управления
- **Window** - окно
- Элементы управления содержимым с заголовками **HeaderedContentControl** : **ContentControl**
  - **GroupBox** – группирующая рамка с заголовком
  - **TabItem** – содержимое в **TabControl**
  - **Expander** – представляет собой раскрывающийся элемент с заголовком

Label

- По сути представляет собой лишь контейнер для содержимого
  - <Label>Это метка</Label>
- Ярлык быстрого доступа
  - <Label >_Это Label</Label>
  - При нажатии **Alt**: Это **Label** (Э подчеркнется)
  - При нажатии клавиш **Alt+Э** (фокус получит элемент указанный в свойстве **Target**)
    - <Label Target="{Binding ElementName=textbox1}“>_Это Label</Label>
    - <TextBox Name="textbox1”/>

Button

- Событие **Click** – нажатие кнопки
- Свойство **IsDefault** – кнопка по умолчанию. Если **IsDefault** = **true**, нажатие **Enter** приведет к нажатию этой кнопки (**Click**).
- Свойство **IsCancel** – кнопка по умолчанию. Если **IsCancel** = **true**, нажатие **Esc** приведет к нажатию этой кнопки (**Click**).
- Также может иметь клавиши быстрого доступа

```
<Button IsDefault="True">_Ok</Button>
<Button IsCancel="True">_Cancel</Button>
```

  - При нажатии **Alt+O** или **Alt_C** – будет вызвано соответствующее событие **Click**
- Свойство **Command** – прикрепление команды, вызывающейся при нажатии на кнопку

CheckBox

- Наследник от класса **ToggleButton : ButtonBase**
- Свойство **IsChecked** типа **bool?**
  - Тринарная логика
  - По умолчанию **IsChecked = null** (не определенное значение)
- Имеет событие **Click**

RadioButton

- Наследник от класса **ToggleButton : ButtonBase**
- Имеет событие **Click**
- Свойство **IsChecked** типа **bool?**
- Можно группировать.
- В группе может быть выбран только один **RadioButton**. При выборе элемента с остальных автоматически снимается выборка
- Все элементы **RadioButton** в одном контейнере автоматически помещаются в одну группу.
- Можно группировать самостоятельно (в одном контейнере) задав свойство **GroupName**
  - <RadioButton GroupName="Gr1"/>
  - <RadioButton GroupName="Gr1"/>
  - <RadioButton GroupName="Gr2"/>
  - <RadioButton GroupName="Gr2"/>

ToolTip

- Всплывающая подсказка
- Не получает фокус
- Почти все элементы управления имеют свойство **ToolTip** (наследуется от **FrameworkElement**), в которое может быть помещен объект элемент **ToolTip**

```
<Button ToolTip="Это подсказка"/>
<Button>
    <Button.ToolTip>
        <Image Source="happy.jpg"/>
    </Button.ToolTip>
</Button>
```

Специализированные элементы управления

- Нет свойства **Content**
- Специализированы в отношении содержимого
- **TextBlock**
- **Image**
- **TextBox**
- **ProgressBar**
- **Slider**
- **Calendar**
- **DatePicker**

TextBlock

- Отображение текста
- Содержимое – текст, содержит свойство **Text**
  - XAML: <TextBlock Text="Текстик"/>
  - XAML: <TextBlock>Текстик</TextBlock>
  - C#: textblock1.Text="Текстик";
- По умолчанию использует шрифт контейнера

```
<TextBlock FontSize="16"
           FontWeight="Bold"
           FontStyle="Italic"
           FontStretch="Expanded">
    Текстик
</TextBlock>
```

Image

- Представляет картинку (изображение)
- Свойство **Source** типа **System.Windows.Media.ImageSource**
- Можно задать с помощью **URI**
  - <Image Source="d:\Yura\Камин.jpg"/>
- Свойство **Stretch** – Сжатие / растяжение картинки:
  - **None** – баз сжатия, обрезается
  - **Fill** – заполняет элемент управления
  - **Uniform** – масштабирование, сохранение пропорций, содержимое не обрезается
  - **UniformToFill** - – масштабирование, сохранение пропорций, содержимое обрезается

TextBox

- Отображает и позволяет редактировать текст
- Содержит только текст. Свойство **Text**.
- Свойство **IsReadOnly** = **true** – только чтение текста пользователем
- Поддерживает переносы по словам. Свойство **TextWrapping**:
  - **None** – нет переноса
  - **Wrap** - перенос текста на другую строчку
  - **WrapWithOverflow** - перенос текста на другую строчку с возможностью выхода за границы **TextBox**
- Свойства полосы прокрутки:
  - **VerticalScrollBarVisibility**
  - **HorizontalScrollBarVisibility**
  - Значения: **Visible, Auto, Hidden**
- Поддерживает **Copy/Paste**
- Поддерживает проверку орфографии

ProgressBar

- Отображает прогресс выполнения длительной задачи
- Свойства:
  - **Maximum**, **Minimum** – макс. и мин. Значения элемента
  - **Value** – Значение прогресса процесса
  - **Orientation** – Вертикальное или горизонтальное расположение
  - **IsEnabled** – вкл./выкл. элемента

Slider

- Графическое изменение значения
- Свойства:
  - **Maximum**, **Minimum**
  - **Value** – текущее значение
  - **Orientation**
  - **LargeChange**, **SmallChange**
  - **TickFrequency** – отметки на линейке прокрутки
- Событие **ValueChanged** при каждом изменение значения **Value**

Демонстрации

Примеры элементов управления

Items Controls

Элементы управления списком

Элементы управления списком

- Отображение множества произвольных элементов
- Позволяют выбрать один из элементов
- **ListBox**
- **ComboBox**
- **TreeView**
- **Menu**, **ContextMenu**
- **ToolBar**
- **ListView**
- **DataGrid**

- Основные свойства
  - **Items**, **ItemsSource** – коллекция содержащихся элементов.

ListBox

- Отображает список элементов
- Как привило отображает список элементов управления содержимым **ListBoxItem**

```
<ListBox>
        <ListBoxItem>This</ListBoxItem>
        <ListBoxItem>is</ListBoxItem>
        <ListBoxItem>a</ListBoxItem>
        <ListBoxItem>List</ListBoxItem>
        <Button>Кнопочка</Button>
</ListBox>
```

- Может содержать элементы любых типов

```
<ListBox >
    <CheckBox>Option 1</CheckBox>
    <CheckBox>Option 2</CheckBox>
    <CheckBox>Option 3</CheckBox>
</ListBox>
```

ListBox

- Автоматически складывает в стопу
- Автоматически добавляет полосу прокрутки
- По умолчанию позволяет выбрать только 1 элемент.
  - **ListBox.SelectedIndex**
  - **ListBox.SelectedItem**
  - **ListBoxItem.IsSelected**
- Множественный выбор. Свойство **SelectionMode**
  - **Single**
  - **Multiple**
  - **Extended** – выбор смежных элементов (**Shift**, **Ctrl**)
- **ListBox.SelectedItems** – множество выбранных элементов

ComboBox

- Аналогичен **ListBox.** Может содержать элементы любых типов
- Отличие лишь в оформлении. Представляет собой выпадающий список
- При выборе элемента отображается его строковое содержимое.
- Если содержимое элемент управления содержимым, то отображается его свойство **Context.**
- Отображаемый текст доступен по свойству **Text**
- Пользователь может редактировать текст
- **IsReadOnly** – пользователю текст доступен только для чтения
- **IsEditable** – можно ли редактировать текст
- **IsDropDownOpen** – программно открыть список

TreeView

- Аналогичен **ListBox**, но представляет данные в виде дерева.
- Содержит элементы **TreeViewItem**
- Свойство **Header** отображается в дереве.

```
<TreeView>
        <TreeViewItem Header="1">
                        <CheckBox>1.1</CheckBox>
                        <CheckBox>1.2</CheckBox>
          </TreeViewItem>
          <TreeViewItem Header="2">
                        <CheckBox>2.1</CheckBox>
                        <CheckBox>2.2</CheckBox>
        </TreeViewItem>
</TreeView>
```

ListView

- Аналогичен **ListBox**, но представляет данные в виде специальном виде. Обычно в виде простой легкой таблицы (**GridView**).
- Свойство **View** задает внешний вид списка. Обычно это **GridView**
- Содержит элементы **ListViewItem**

- **GridView** располагает данные в виде таблицы.
- Колонки задаются с помощью **GridViewColumn**

```
<ListView>
  <ListView.View>
    <GridView>
      <GridViewColumn Header="Имя"
                      DisplayMemberBinding="{Binding Path=Name}"/>
      <GridViewColumn Header="Фамилия"
                      DisplayMemberBinding="{Binding Path=LastName}"/>
    </GridView>
  </ListView.View>

  <local:Employee Name="Вася" LastName="Петров"/>
  <local:Employee Name="Петя" LastName="Серый"/>
</ListView>
```

DataGrid

- Отображение данных в виде таблицы
- Свойства
  - ItemsSource – задает коллекцию данных
  - Columns – коллекция определений колонок
    - Типы колонок:
    - DataGridTextColumn – колонка, содержащая текстовые данные
    - DataGridCheckBoxColumn – колонка с галочками
    - DataGridHyperlinkColumn – колонка с гиперссылками
    - DataGridComboBoxColumn – колонка, содержащая **ComboBox**
    - DataGridTemplateColumn – произвольная колонка. Отображение и редактирование данных задаются произвольными шаблонами

Menu, ContextMenu

- **IsMainMenu = true**, привязывает клавиши **Alt** или **F10**
- Содержат коллекцию **MenuItem**
- **MenuItem** содержит:
  - **Header** – отображаемый текст меню
  - **Command** – команда для меню
  - **Icon** – иконка
  - **IsChecked**
  - **IsEnabled**
  - **Items** – коллекция вложенных меню
- Можно назначить клавишу быстрого доступа **_Name**
- **Separator** – разделительная полоса в меню

ToolBar

- Панель элементов
- **Items** – содержит коллекцию элементов

```
<ToolBar>
        <Button>1</Button>
        <Button>2</Button>
        <Button>3</Button>
        <CheckBox>Галочка</CheckBox>
</ToolBar>
```

- При нехватки места на панели элементы перемещаются в меню **Overflow**
- **OverflowMode:**
  - **Always** – всегда отображать и в меню **Overflow**
  - **AsNeeded** – только когда все не умещается
  - **Never** – никогда

Демонстрации

Примеры элементов управления

Диалоговые окна

- **MessageBox**
  - показ модальных диалоговых окон
  - **MessageBox.Show**(this, "Message", "Title", MessageBoxButton.YesNoCancel, MessageBoxImage.Question, MessageBoxResult.Yes);

- **OpenFileDialog**, **SaveFileDialog**
  - Диалоговые окна открытия и сохранения файлов
  - **Filter** - Позволяет задать фильтры файлов
  - **Multiselect** – множественный выбор
  - **CheckFileExists** , **CheckPathExists**– проверка на существование файла, пути
  - **InitialDirectory** – папка при открытии окна
  - **CreatePrompt**, **OverwritePrompt** – показывать запросы пользователю на создание или перезапись файла (для SaveFileDialog)
  - **ValidateNames** – проверять валидность имен файлов
  - **ShowDialog**() – показывает окно для открытия/сохранения файла(-ов). Возвращает **true**, если пользователь нажал **Ok**.

```
OpenFileDialog opf = new OpenFileDialog();
opf.Filter = "Word Documents|*.doc|Excel Worksheets|*.xls|PowerPoint Presentations|*.ppt|Office
Files|*.doc;*.xls;*.ppt|All Files|*.*";
opf.CheckFileExists = true;
 if (opf.ShowDialog() == true)
 {
        Open(opf.FileName);
        // или    Stream s = opf.OpenFile();
}
```

Демонстрации

Примеры диалоговых окон