WPF: класс Aplication

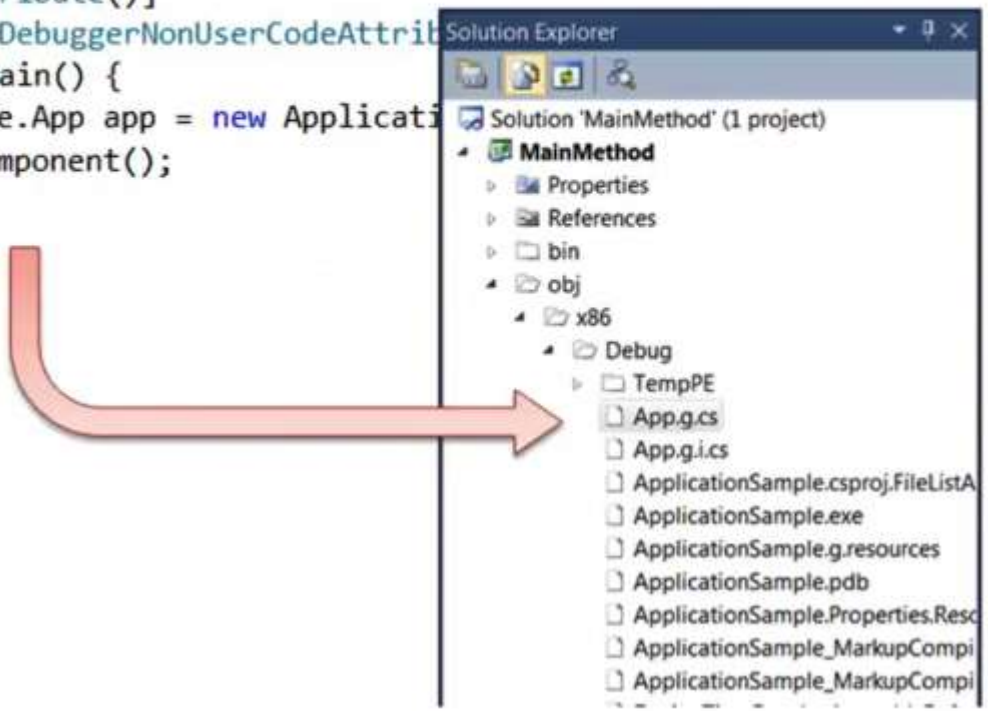
События класса Application

- Start
- Exit
- SessionEnding
- Activated
- Deactivated
- DispatcherUnhandledException

WPF: static void Main ()

Точка входа в приложение

```
56      /// <summary>
57      /// Application Entry Point.
58      /// </summary>
59      [System.STAThreadAttribute()]
60      [System.Diagnostics.DebuggerNonUserCodeAttrib
61      public static void Main() {
62          ApplicationSample.App app = new Applicati
63          app.InitializeComponent();
64          app.Run();
65      }
```



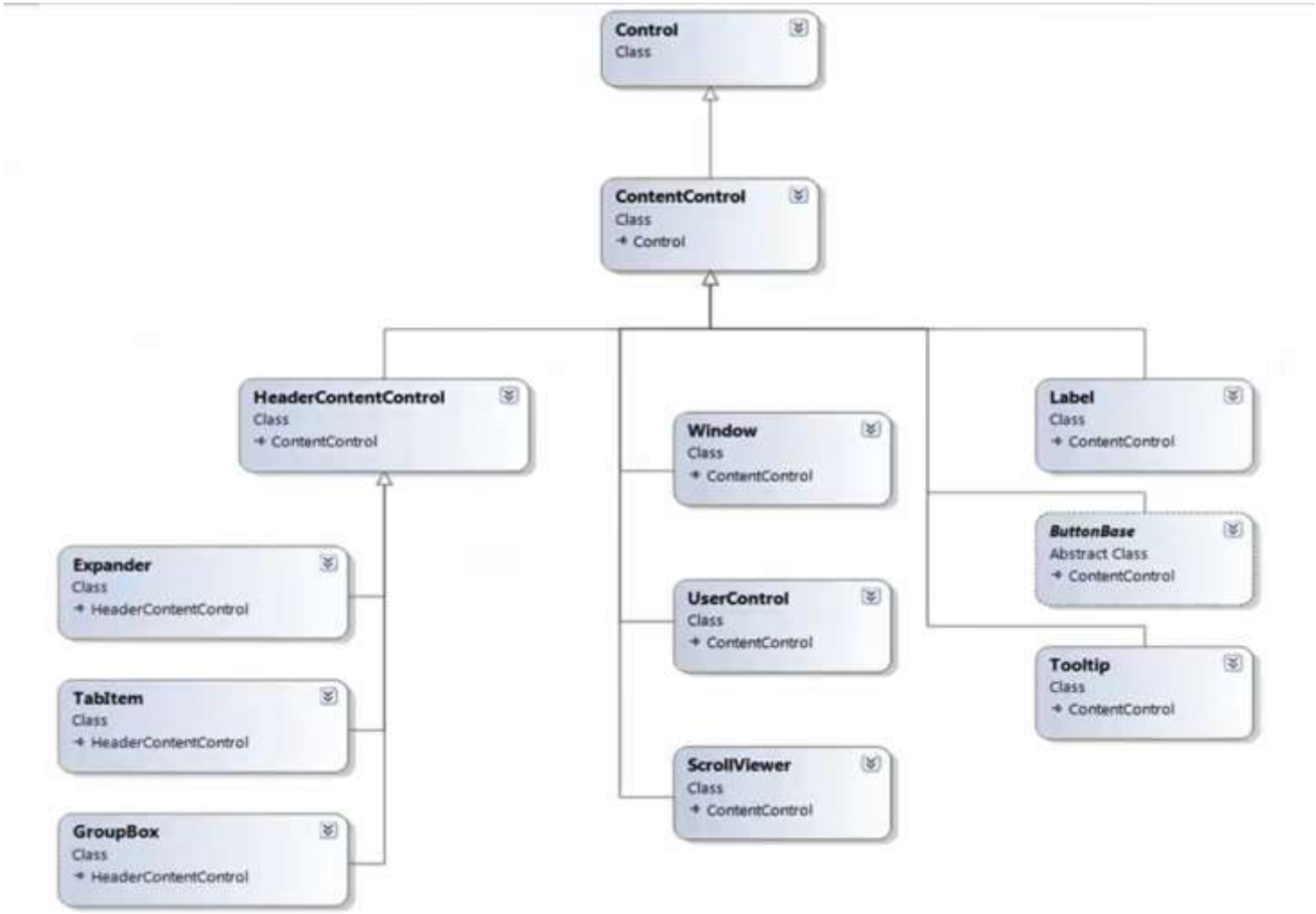
WPF: содержимое

Свойство Content

Класс `ContentControl` добавляет свойство `Content`, которое поддерживает любой тип объекта в качестве содержимого. Все типы, которые можно присвоить свойству `Content` можно разбить на две группы:

- **Типы НЕ производные от `UIElement`.** Объект, установленный в качестве содержимого, будет отображаться как строка, которую возвращает метод `ToString()`
- **Типы производные от `UIElement`.** Объект, установленный в качестве содержимого, будет визуализироваться посредством метода `UIElement.OnRender()`

WPF: иерархия управления содержимым



WPF: элементы управления содержимым

Label

Класс Label является простейшим элементом управления содержимым. Как и любой другой элемент управления содержимым он принимает одиночную порцию содержимого, которая размещается внутри него. Отличительной чертой элемента Label является его поддержка мнемонических комманд—нажатий клавиш, которые передают фокус соответствующему элементу управления.

ButtonBase

Класс ButtonBase содержит лишь несколько членов. Он определяет событие Click и добавляет поддержку комманд, которые позволяют подключать кнопки к высокоуровневым задачам приложений.

WPF: элементы управления содержимым

ToolTip

Представляет элемент управления, который создает всплывающее окно, отображающее сведения для элемента в интерфейсе. Например, можно использовать объект ToolTip для предоставления имени Button или ToolBar в объекте ToolBarTray. Содержимое элемента управления ToolTip может меняться от простой текстовой строки до более сложного содержимого, такого как объект StackPanel, который содержит встроенный текст и изображения. Содержимое объекта ToolTip не может получать фокус.

ScrollViewer

Класс ScrollViewer нужен для того чтобы добавить полосу прокрутки в окно, которая необходима при размещении большого объема содержимого в ограниченную область.

WPF: элементы управления содержимым

UserControl

Если необходимо создать новый элемент управления, то самый простой способ создать класс, производный от UserControl.

Window

Предоставляет возможность создания, настройки отображения и управления времени существования окон и диалоговых окон.

WPF: элементы управления содержимым

HeaderContentControl

HeaderContentControl это просто контейнер с содержимым и заголовком. От класса HeaderContentControl порождены классы GroupBox, TabItem и Expander.

GroupBox

Класс GroupBox используется для группировки небольшого колличества взаимосвязанных элементов управления, таких как кнопки переключателя. В классе GroupBox нет никаких встроенных функций, поэтому его можно применять где угодно.

WPF: элементы управления содержимым

TabItem

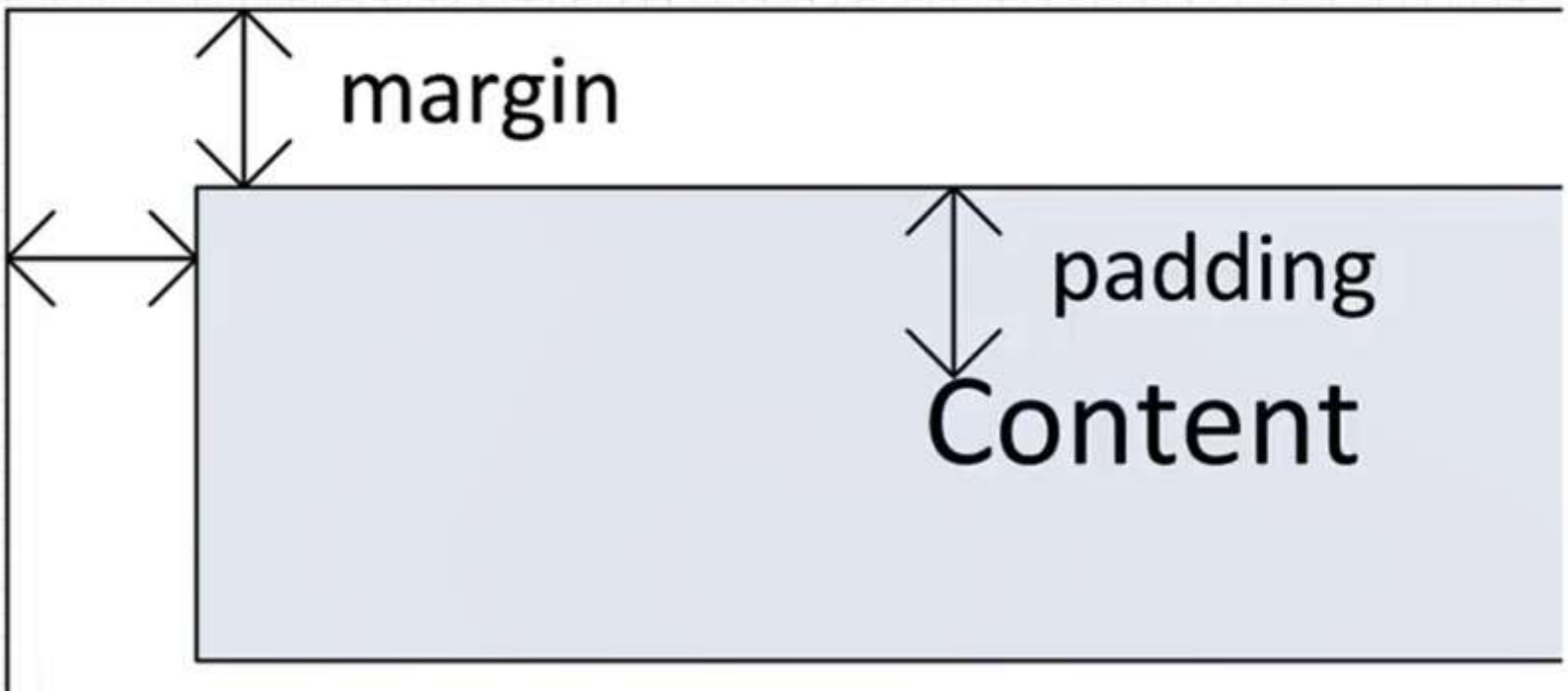
Объекты TabItem представляют собой страницы в элементе TabControl. Свойство IsSelected указывает видима ли данная вкладка в TabControl.

Expander

Класс Expander содержит область содержимого, которую пользователь может показать или скрыть, щелкнув на кнопочке со стрелкой.

WPF: выравнивание содержимого

Свойства Padding и Margin



WPF:декоратор Border

Свойства

Свойство	Описание
Background	Задает фон, который отображается за всем содержимым в рамке.
BorderBrush и BorderThickness	Цвет и ширина рамки
CornerRadius	Закругление контура по углам. Чем больше значение, тем более выразительней будет виден эффект закругления.
Padding	Добавление пустого пространства между содержимым и рамкой.

WPF: свойства зависимостей и маршрутизируемые события

Определение свойства зависимостей.

```
public static DependencyProperty DataProperty
```

Сначала нужно определить объект, который будет представлять свойство. Это экземпляр класса DependencyProperty. Информация о свойстве должна быть доступна постоянно и даже другим классам по этой причине объект DependencyProperty следует определить как статическое поле.
По соглашению, поле, представляющее свойство зависимости, имеет имя обычного свойства плюс слово **Property** в конце.

WPF: регистрация свойства зависимости

Dependency Property

Определение объекта DependencyProperty является лишь первым шагом. Чтобы его можно было задействовать, необходимо зарегистрировать свойство зависимости в WPF. Это нужно сделать до использования данного свойства в коде, поэтому определение должно быть выполнено в статическом конструкторе связанного класса.

```
static MyFirstControl()
{
    DataProperty = DependencyProperty.Register("Data", typeof(int), typeof(MyFirstControl));
}
```

WPF: чтение и изменение свойства зависимости

Dependency Property

```
public int Data
{
    get
    {
        return (int)GetValue(DataProperty);
    }
    set
    {
        SetValue(DataProperty, value);
    }
}
```

Методы доступа в свойстве, которое является оболочкой для свойства зависимостей, должны использовать методы `SetValue()` и `GetValue()`, определенные в классе `DependencyObject`, для чтения и изменения значения свойства зависимостей.

WPF: маршрутизируемые события

Routed Event

Маршрутизируемые события – это события проходящие определенный маршрут по дереву элементов управления. Маршрутизируемые события позволяют обработать событие в одном элементе (например в метке), хотя оно возникло в другом (например в изображении внутри метки).

Типы событий:

Tunnel – распространяются от корневого элемента дерева до элемента инициирующего событие.

Bubble – распространяются от элемента который инициировал событие к корню дерева

Direct – событие происходит только в конкретном элементе управления

WPF: маршрутизируемые события

Routed Event

При определении маршрутизируемого события используется следующее правило именования – <Имя события>Event .

```
public static RoutedEvent MyButtonClickEvent;
```

При регистрации события указывается маршрут события, тип делегата события и класс владеющий данным событием.

```
MyButtonClickEvent = EventManager.RegisterRoutedEvent("MyButtonClick",
    RoutingStrategy.Direct,
    typeof(RoutedEventHandler),
    typeof(ExtraButton));
```

WPF: обработчик события

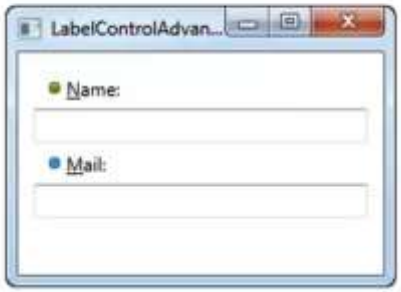
Routed Event

Удаление и добавление обработчиков события производится с помощью методов AddHandler() и RemoveHandler() определенных в классе FrameworkElement

```
public event RoutedEventHandler MyButtonClick
{
    add { AddHandler(MyButtonClickEvent, value); }
    remove { RemoveHandler(MyButtonClickEvent, value); }
}
```

WPF: классические элементы управления

Класс Label



Класс Label – предоставляет текстовую метку для элемента управления и обеспечивает возможность фокусировки на элементах управления после нажатия на горячие клавиши.

Для фокусировки используются свойства Target и Content.

Target – элемент, к которому будет происходить привязка.

Content - текст и горячая клавиша (в комбинации с Alt).

WPF: классические элементы управления

Класс TextBox

Класс TextBox– представляет элемент управления, который может использоваться для отображения или изменения неформатированного текста. Свойства:

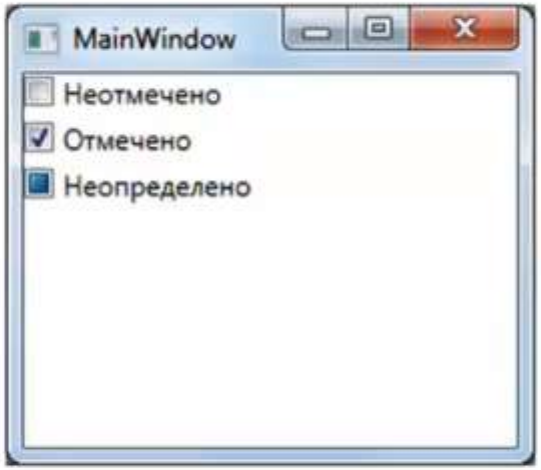
SpellCheck.IsEnabled="True" - проверка ошибок.

SelectionChanged - событие при смене выделенного текста.

TextWrapping="Wrap" - текст, который не поместился по ширине элемента будет перенесен.

WPF: классические элементы управления

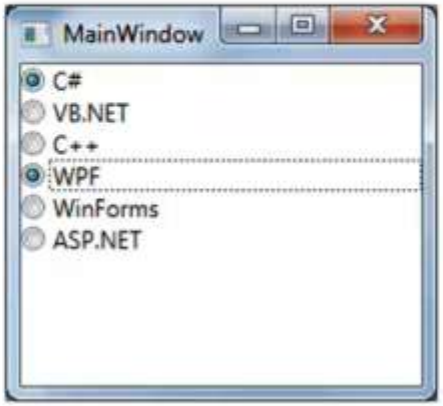
Класс CheckBox



Класс CheckBox – представляет элемент управления флажок, который пользователь может устанавливать и снимать.

WPF: классические элементы управления

Класс RadioButton



Класс RadioButton – представляет переключатель, который пользователь может устанавливать (выбирать), но не снимать (отменять выбор). Свойство IsChecked элемента RadioButton устанавливается, когда пользователь щелкает этот элемент, но очистить свойство можно только программным способом. RadioButton по умолчанию группируется контейнерами, но также,
можно группировать эти элементы управления с помощью свойства GroupName.

WPF: классические элементы управления

Класс ToolTip

Класс ToolTip – представляет элемент управления, который создает всплывающее окно, отображающее сведения для элемента в интерфейсе.

WPF: классические элементы управления

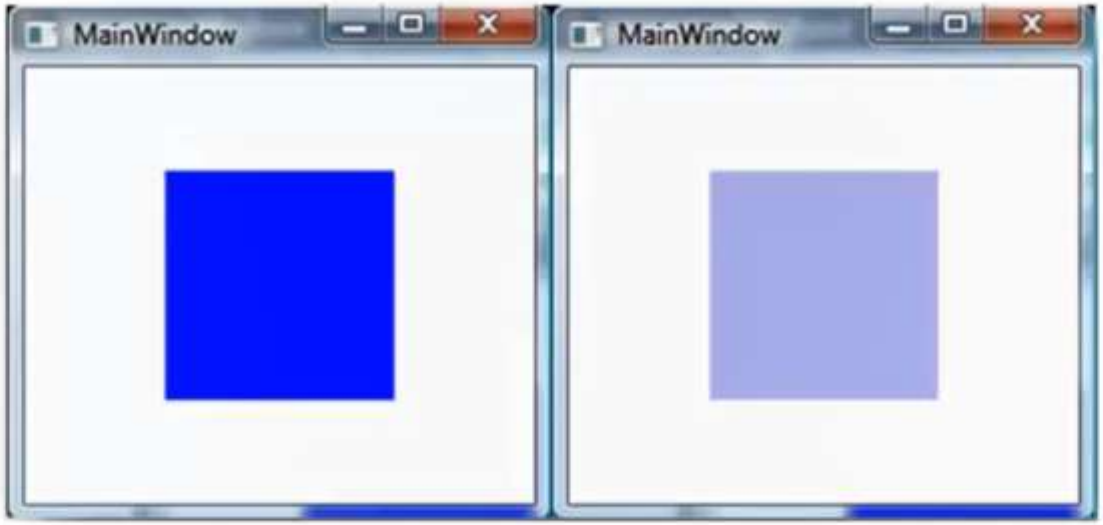
Класс PopUp

Класс PopUp – представляет всплывающее окно с содержимым. Отличия от ToolTip:

1) Имеет свойство PopupAnimation.
2) Popup может принимать фокус, таким образом, в него можно помещать другие элементы управления.

WPF: классические элементы управления

Свойство Opacity



Opacity - свойство устанавливающее непрозрачность элемента. 1 - полностью непрозрачный, 0 - полностью прозрачный.

WPF: свойства окна

AllowsTransparency – позволяет использовать фон с прозрачностью.
Icon – путь к файлу с иконкой для приложения.
Top и **Left** – расстояние между углом окна и краями экрана.
ResizeMode – перечисление указывающее на то можно ли пользователю изменять размеры окна.
RestoreBounds – свойство извлекает информацию о границах окна, до изменения размеров.
ShowInTaskBar – указывает на то будет ли окно видимо в панели задач.
SizeToContent – окно увеличивается по мере увеличения размеров контента.
Title – заголовок окна.
Topmost – если значение true то окно будет отображаться поверх остальных окон.
WindowStartupLocation – позиция в которой окно будет отображено.
WindowState – состояние окна: свернуто, развернуто или обычное состояние.
WindowsStyle - стиль рамки окна.

WPF: типы окон

- Модальное (запускается с помощью метода `ShowDialog()`) – доступ к родительскому окну запрещен пока открыто модальное окно.
- Немодальное (запускается с помощью метода `Show()`) – доступ к родительскому окну открыт.

WPF: диалоговые окна

OpenFileDialog & SaveFileDialog

Для использования стандартных диалоговых окон, следует подключить пространство имен Microsoft.Win32

Для отображения этих диалоговых окон используется метод ShowDialog(). Метод возвращает true если пользователь подтвердил действие, например, нажал кнопку сохранить и false если пользователь нажал отмена.

WPF: события окна

ContentRendered – возникает сразу же после первой визуализации окна.
Loaded – происходит когда окно полностью инициализировано и готово к взаимодействию.
Activated – возникает когда пользователь переключается на это окно, а также при первой загрузке окна.
Deactivated – возникает, когда пользователь переходит на другое окно, а также когда окно закрывается.
Closing – возникает при закрытии окна, позволяет отменить операцию закрытия.
Closed – возникает после закрытия окна.

WPF: нестандартные окна

1. Задать свойство окна `AllowTransparency = true`
2. Установить свойство окна `WindowsStyle = None`
3. В качестве фона установить картинку с прозрачными областями.

WPF: нестандартные окна

1. Задать свойство окна `AllowTransparency = true`
2. Установить свойство окна `WindowsStyle = None`
3. В качестве фона установить картинку с прозрачными областями.

WPF: нестандартные окна

1. Задать свойство окна `AllowTransparency = true`
2. Установить свойство окна `WindowsStyle = None`
3. В качестве фона установить картинку с прозрачными областями.

WPF: ресурсы проекта

- **Ресурс сборки (assembly resource)** – блок двоичных данных, встроенный в скомпилированную сборку.
- **Ресурс объекта** или **объектный ресурс (object resource)** – это .NET-объект который объявляется в одном месте, а использует в нескольких других.

WPF: ресурсы проекта

Добавление ресурса в проект