

Промышленное программирование

Лекция 10. AvaloniaUI (часть 1: XAML)

- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы
- 3 XAML
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI

- 1 Шаблоны UI-проектов**
- 2 Инфраструктурные классы
- 3 XAML
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI

Подготовка: Avalonia (Visual Studio)

1. Установить расширение [Avalonia для Visual Studio](#)
2. При создании проекта выбрать один из следующих шаблонов:



3. Для UI-отладки после сборки и запуска в самом приложении нажать на F12

Подготовка: Avalonia (для любой IDE)

1. Установить шаблоны Avalonia, если они ещё не установлены:

```
dotnet new -i Avalonia.Templates
```

**2. Перейти в папку, где находится или будет находиться решение
Если файла решения не существует, создать его:**

```
dotnet new sln -n MySolution
```

3. Создать новый проект на основе шаблона:

```
dotnet new avalonia.app -n MyAvaloniaProject
```

4. Добавить проект в решение:

```
dotnet sln add MyAvaloniaProject
```

5. Для сборки и запуска через консоль:

```
cd MyAvaloniaProject  
dotnet run
```

- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы**
- 3 XAML
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI

Аналог object в мире UI

```
namespace Avalonia;  
  
class AvaloniaObject : INotifyPropertyChanged, ...  
{  
    event ... PropertyChanged;  
  
    bool CheckAccess();  
    void VerifyAccess();  
  
    ... // Обеспечение механизма свойств зависимостей  
}
```

Класс приложения

```
namespace Avalonia;

class Application : AvaloniaObject, ...
{
    static Application? Current { get; }

    IApplicationLifetime? ApplicationLifetime { get; set; }

    virtual void Initialize()
    virtual void OnFrameworkInitializationCompleted();

    ...
}

namespace Avalonia.Controls.ApplicationLifetimes;

interface IClassicDesktopStyleApplicationLifetime : ...
{
    // IControlledApplicationLifetime
    event ... Exit;

    Window MainWindow { get; set; }
    IReadOnlyList<Window> Windows { get; }

    ...
}
```

Диспетчер UI-потока

```
namespace Avalonia.Threading;

class Dispatcher : ...
{
    static Dispatcher UIThread { get; }

    // if (Application.Current.ApplicationLifetime is
    //     IControlledApplicationLifetime lifetime)
    //     lifetime.Exit += ...

    bool CheckAccess();
    void VerifyAccess();

    void Post(Action action,
              DispatcherPriority priority = Normal)

    Task InvokeAsync(Action action,
                    DispatcherPriority priority = Normal);
    Task<TResult> InvokeAsync<TResult>(
        Func<TResult> function,
        DispatcherPriority priority = Normal)

    ...
}
```


- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы
- 3 XAML**
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI

Для такой простой формы так много кода?!

```
var loginLabel = new Label {
    Content = "Логин:",
    HorizontalContentAlignment = HorizontalAlignment.Right,
    Width = 60
};
var loginTextBox = new TextBox {
    VerticalAlignment = VerticalAlignment.Center,
    Width = 150
};
var loginPanel = new StackPanel { Orientation = Orientation.Horizontal };
loginPanel.Children.Add(loginLabel);
loginPanel.Children.Add(loginTextBox);

var passwordLabel = new Label {
    Content = "Пароль:",
    HorizontalContentAlignment = HorizontalAlignment.Right,
    Width = 60
};
var passwordTextBox = new TextBox {
    VerticalAlignment = VerticalAlignment.Center,
    Width = 150
};
var passwordPanel = new StackPanel { Orientation = Orientation.Horizontal };
passwordPanel.Children.Add(passwordLabel);
passwordPanel.Children.Add(passwordTextBox);

var rootPanel = new StackPanel { Margin = new Thickness(10) };
rootPanel.Children.Add(loginPanel);
rootPanel.Children.Add(passwordPanel);

var window = new Window { SizeToContent = SizeToContent.WidthAndHeight };
window.Content = rootPanel;
```



XAML: Extensible Application Markup Language

```
<Window x:Class="SimpleWPF.Example"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        SizeToContent="WidthAndHeight">
  <StackPanel Margin="10">
    <StackPanel Orientation="Horizontal">
      <Label Content="Логин:" HorizontalContentAlignment="Right" Width="60" />
      <TextBox VerticalAlignment="Center" Width="150" />
    </StackPanel>
    <StackPanel Orientation="Horizontal">
      <Label Content="Пароль:" HorizontalContentAlignment="Right" Width="60" />
      <TextBox VerticalAlignment="Center" Width="150" />
    </StackPanel>
  </StackPanel>
</Window>
```



Логин:

Пароль:

- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы
- 3 XAML
- 4 Обзор классов: элементы управления**
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI

Базовые классы (1)

```
namespace Avalonia;  
  
class Visual : StyledElement ← Animatable ←  
              AvaloniaObject, ...  
{  
    Rect Bounds { get; protected set; }  
  
    bool IsVisible { get; set; }  
  
    double Opacity { get; set; }  
}
```

Базовые классы (2)

```
namespace Avalonia.Layout;

class Layoutable : Visual, ...
{
    double MinWidth { get; set; }
    double MinHeight { get; set; }
    double MaxHeight { get; set; }
    double MaxWidth { get; set; }
    double Width { get; set; }
    double Height { get; set; }

    HorizontalAlignment HorizontalAlignment { get; set; }
    VerticalAlignment VerticalAlignment { get; set; }

    Thickness Margin { get; set; }
}

struct Thickness : ...
{
    double Left { get; set; }
    double Top { get; set; }
    double Right { get; set; }
    double Bottom { get; set; }
}
```

Базовые классы (3)

```
namespace Avalonia.Input;

class InputElement : Interactive ← Layoutable, ...
{
    event ... Tapped;
    event ... DoubleTapped;

    event ... PointerPressed;
    event ... PointerReleased;
    event ... PointerMoved;
    event ... PointerWheelChanged;

    event ... KeyDown;
    event ... KeyUp;
}
```

Базовые классы (4)

```
namespace Avalonia.Controls.Primitives;

class TemplatedControl : Control ← InputElement, ...
{
    FontFamily FontFamily { get; set; }
    FontWeight FontWeight { get; set; }
    FontStyle FontStyle { get; set; }
    double FontSize { get; set; }
    IBrush Foreground { get; set; }

    IBrush Background { get; set; }
    IBrush BorderBrush { get; set; }
    Thickness BorderThickness { get; set; }

    Thickness Padding { get; set; }
}

namespace Avalonia.Controls;

class ContentControl: TemplatedControl
{
    object Content { get; set; }
}
```


Окна

```
namespace Avalonia.Controls;

class Window : BaseWindow ← TopLevel ← ContentControl
{
    WindowIcon Icon { get; set; }
    string Title { get; set; }

    WindowBase Owner { get; protected set; }
    WindowStartupLocation WindowStartupLocation { get; set; }
    WindowState WindowState { get; set; }

    bool CanResize { get; set; }
    SizeToContent SizeToContent { get; set; }

    void Show();
    void Show(Window parent);

    Task ShowDialog(Window owner)
    Task<TResult> ShowDialog<TResult>(Window owner)

    void Close();
    void Close(object dialogResult);
}
```

Подписи

[Label Reference](#)


```
<Label Content="Мой текст:" />
```

Мой текст:

Текстовые поля

[TextBox Reference](#)

```
<TextBox MaxLength="10" Text="Мой текст" />
```



Мой текст

Кнопки

[Button Reference](#)

```
<Button Content="Нажми меня" Click="OnClick" />
```

A rectangular button with a light gray background and rounded corners. The text "Нажми меня" is centered on the button in a dark gray font.

Флажки

[CheckBox Reference](#)

```
<CheckBox Content="Мой флажок" IsChecked="True" />
```



Переключатели

RadioButton

```
<RadioButton GroupName="A" Content="Вариант 1" />  
<RadioButton GroupName="A" Content="Вариант 2" IsChecked="True" />
```

☐ Вариант 1

☒ Вариант 2

Списки

[ListBox Reference](#)

```
<ListBox SelectedIndex="2">  
  <ListBoxItem Content="Первый элемент" />  
  <ListBoxItem Content="Второй элемент" />  
  <ListBoxItem Content="Третий элемент" />  
</ListBox>
```

Первый элемент

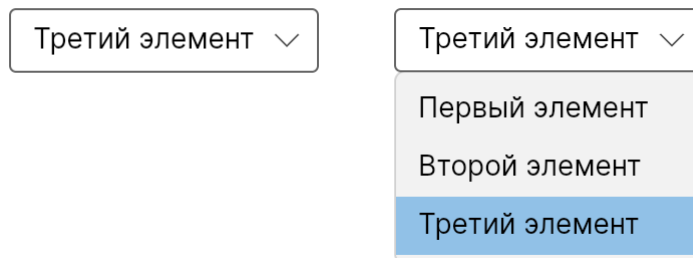
Второй элемент

Третий элемент

Выпадающие списки

ComboBox Reference

```
<ComboBox SelectedIndex="2">  
  <ComboBoxItem Content="Первый элемент" />  
  <ComboBoxItem Content="Второй элемент" />  
  <ComboBoxItem Content="Третий элемент" />  
</ComboBox>
```



И так далее...

1. [Стандартные элементы управления](#)
2. [Список сторонних библиотек](#)

- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы
- 3 XAML
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки**
- 6 Генерация клиентского кода OpenAPI

Panel

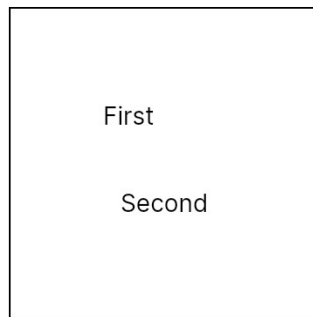
Базовый класс для всех диспетчеров компоновки

```
namespace Avalonia.Controls;  
  
class Panel : Control, ...  
{  
    Controls Children { get; }  
  
    ...  
}
```

Canvas

[Canvas Docs](#)

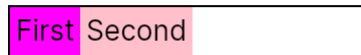
```
<Canvas>  
  <Label Content="First" Canvas.Left="50" Canvas.Top="50" />  
  <Label Content="Second" Canvas.Left="60" Canvas.Top="100" />  
</Canvas>
```



StackPanel

[StackPanel Docs](#)

```
<StackPanel Orientation="Horizontal">  
  <Label Background="Magenta" Content="First" />  
  <Label Background="Pink" Content="Second" />  
</StackPanel>
```



Grid

[Grid Docs](#)

```
<Grid RowDefinitions="Auto,Auto" ColumnDefinitions="Auto,Auto">  
  <Label Grid.Row="0" Grid.Column="0" Content="Логин:" />  
  <Label Grid.Row="1" Grid.Column="0" Content="Пароль:" />  
  <Panel Grid.Row="0" Grid.Column="1" Grid.RowSpan="2" Background="Pink" Width="50"/>  
</Grid>
```

Логин:	
Пароль:	

DockPanel

[DockPanel Docs](#)

```
<DockPanel>  
  <Rectangle DockPanel.Dock="Top" Height="30" Fill="Pink" />  
  <Rectangle DockPanel.Dock="Left" Width="40" Fill="Yellow" />  
  <Rectangle DockPanel.Dock="Left" Width="20" Fill="Blue" />  
  <Rectangle Fill="Gray" />  
</DockPanel>
```



И так далее...

[Avalonia: Layout](#)

- 1 Шаблоны UI-проектов
- 2 Инфраструктурные классы
- 3 XAML
- 4 Обзор классов: элементы управления
- 5 Обзор классов: диспетчеры компоновки
- 6 Генерация клиентского кода OpenAPI**

Генерация клиентского кода OpenAPI

0. Предварительно сохранить swagger.json-файл (или запустить сервер)
1. ПКМ по проекту, Add → Connected Service, выбрать OpenAPI
2. В окне “Add new OpenAPI service reference”:
 - а) указать путь к сохранённому swagger.json (или URI у запущенного сервера)
 - б) факультативно предоставить пространство имён (например, OpenAPI)
 - в) факультативно предоставить имя клиентского класса (например, UserWallClient)

Для явного указания имен методов на сервере можно использовать свойство Name:

```
[HttpGet(Name = "GetUsers")]  
public async Task<IEnumerable<UserDto>> Get();
```