

.Net MAUI

ВВЕДЕНИЕ В XAML

Проект .Net MAUI

.NET MAUI

Проект .Net MAUI

Проект .NET MAUI изначально содержит:

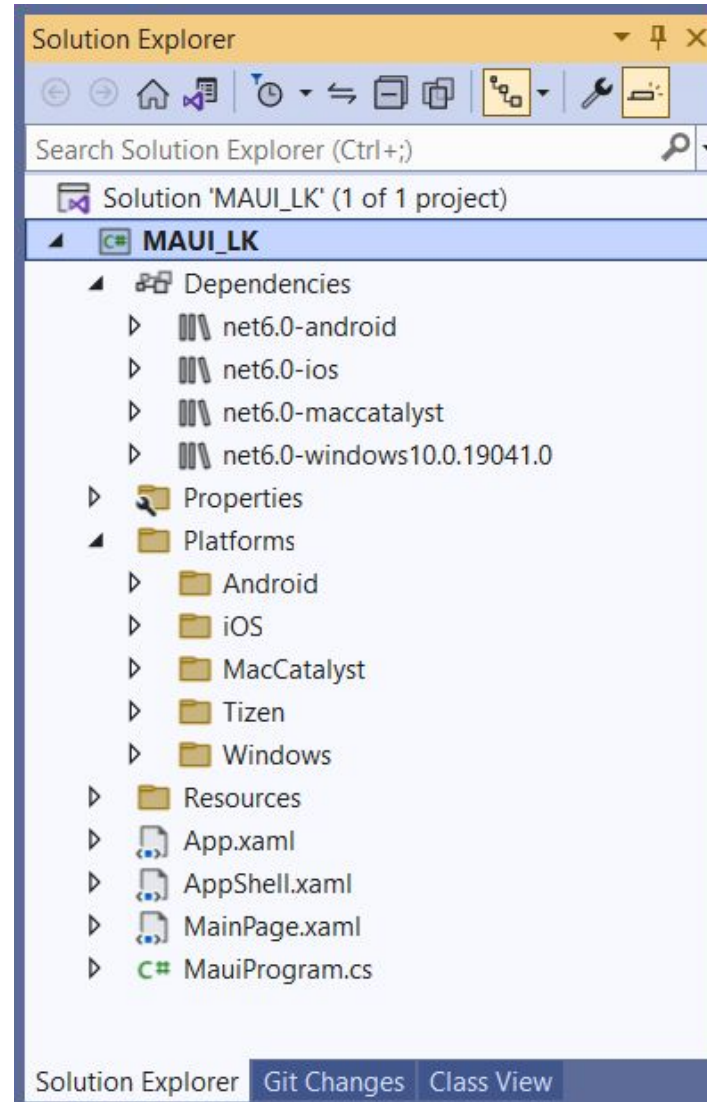
- Файл MauiProgram.cs, содержащий код для создания и настройки объекта Application.
- Файлы App.xaml и App.xaml.cs, предоставляющие ресурсы пользовательского интерфейса и создающие начальное окно для приложения.
- Файлы AppShell.xaml и AppShell.xaml.cs, которые определяют начальную страницу для приложения и обрабатывают регистрацию страниц для маршрутизации навигации.
- Файлы MainPage.xaml и MainPage.xaml.cs, которые определяют макет и логику пользовательского интерфейса для страницы, отображаемой по умолчанию в начальном окне.

Проект .Net MAUI

При необходимости вы можете добавить в приложение дополнительные страницы и создать дополнительные классы для реализации бизнес-логики, требуемой приложением.

Проект .NET MAUI также содержит набор ресурсов приложения по умолчанию, таких как изображения, значки и шрифты, а также код начальной загрузки по умолчанию для каждой платформы.

Проект .Net MAUI



Проект .Net MAUI

Класс **App** представляет приложение .NET MAUI в целом. Он наследует набор поведения по умолчанию от `Microsoft.Maui.Controls.Application`.

Именно класс **App** создается и загружается кодом начальной загрузки для каждой платформы. Конструктор класса **App**, в свою очередь, обычно создает экземпляр класса **AppShell** и присваивает его свойству `MainPage`. Именно этот код управляет первым экраном, который видит пользователь, через то, что определено в **AppShell**.

Проект .Net MAUI

Класс **App** также содержит:

- Методы обработки событий жизненного цикла, в том числе когда приложение отправляется в фоновый режим (то есть когда оно перестает быть активным приложением).
- Способы создания новых окон для приложения. Приложение .NET MAUI по умолчанию имеет одно окно, но вы можете создавать и запускать дополнительные окна, что полезно в настольных и планшетных приложениях.

Проект .Net MAUI

```
protected override void OnStart()  
{  
    base.OnStart();  
}
```


Проект .Net MAUI

Файл **App.xaml** определяет ресурсы приложения, которые приложение будет использовать в макете XAML.

Ресурсы по умолчанию расположены в папке «Ресурсы» и определяют цвета и стили приложения по умолчанию для каждого встроенного элемента управления .NET MAUI.

Проект .Net MAUI

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Resources/Styles/Colors.xaml" />

<ResourceDictionary Source="Resources/Styles/Styles.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

Проект .Net MAUI

Shell упрощает разработку приложений, предоставляя основные функции, которые требуются большинству приложений, в том числе:

- Единое место для описания визуальной иерархии приложения.
- Общий пользовательский интерфейс навигации.
- Схема навигации на основе URI, позволяющая переходить на любую страницу приложения.
- Встроенный обработчик поиска.

Проект .Net MAUI

В приложении .NET MAUI Shell визуальная иерархия приложения описывается в классе, который является подклассом класса Shell. Этот класс может состоять из трех основных иерархических объектов:

- FlyoutItem или TabBar. FlyoutItem представляет один или несколько элементов во всплывающем меню и должен использоваться, когда шаблон навигации для приложения требует всплывающего меню. TabBar представляет нижнюю панель вкладок и должен использоваться, когда шаблон навигации для приложения начинается с нижних вкладок и не требует всплывающего меню.
- Вкладка, представляющая сгруппированное содержимое, навигация по которым осуществляется с помощью нижних вкладок.
- ShellContent, который представляет объекты ContentPage для каждой вкладки.

Проект .Net MAUI

Объекты, описанные в Shell, представляют не какой-либо пользовательский интерфейс, а **организацию визуальной иерархии приложения**. Shell возьмет эти объекты и создаст пользовательский интерфейс для навигации по содержимому.

Проект .Net MAUI

Pages являются корнем иерархии пользовательского интерфейса в .NET MAUI внутри **Shell**.

Класс страницы является производным от **ContentPage**, который является самым простым и наиболее распространенным типом страницы. **ContentPage** просто отображает свое содержимое. .NET MAUI также имеет несколько других встроенных типов страниц, в том числе следующие:

- **TabbedPage**: это корневая страница, используемая для навигации по вкладкам. Страница с вкладками содержит дочерние объекты страницы; по одному на каждую вкладку.
- **FlyoutPage**: эта страница позволяет реализовать презентацию в стиле «основной/подробный». Всплывающая страница содержит список элементов. При выборе элемента появляется представление, отображающее сведения об этом элементе.

Проект .Net MAUI

ContentPage обычно содержит представление (**View**). Представление позволяет извлекать и представлять данные определенным образом. Представлением по умолчанию для страницы содержимого является **ContentView**, в котором элементы отображаются как есть.

Проект .Net MAUI

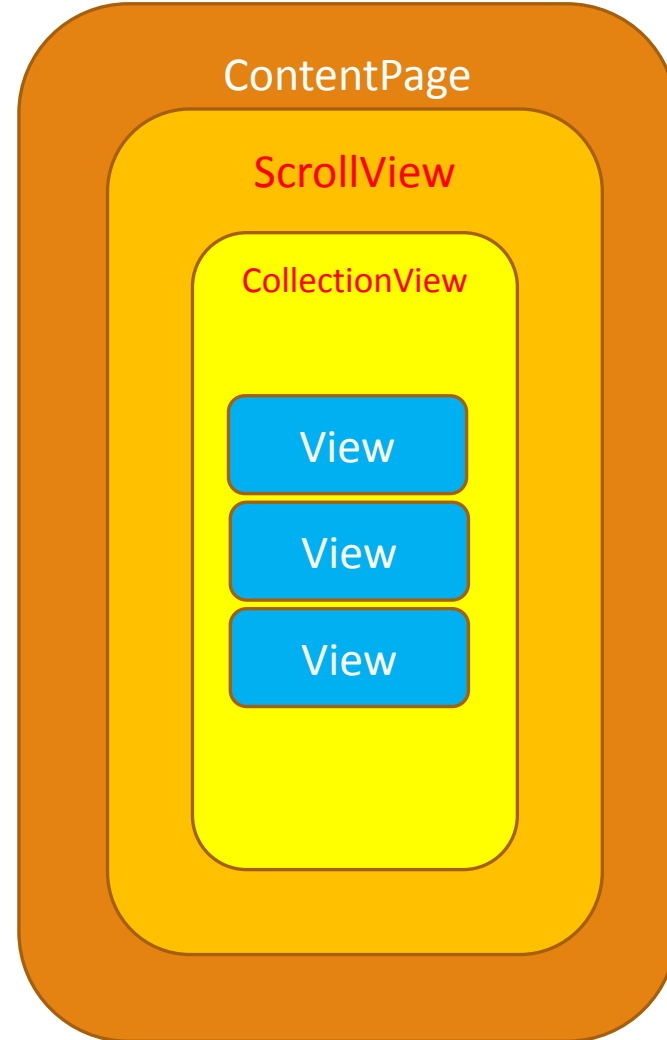
Примеры других типов View:

ScrollView позволяет отображать элементы в прокручиваемом окне; если вы уменьшите окно, вы можете прокручивать вверх и вниз, чтобы отобразить элементы.

CarouselView — это прокручиваемое представление, которое позволяет пользователю пролистывать коллекцию элементов.

CollectionView может извлекать данные из именованного источника данных и представлять каждый элемент с использованием шаблона в качестве формата.

Проект .Net MAUI



Введение в XAML

ОБЩАЯ ИНФОРМАЦИЯ

Введение в XAML

XAML (англ. eXtensible Application Markup Language) — расширяемый язык разметки для приложений

Это декларативный язык разметки, созданный Microsoft, и который можно использовать для создания пользовательского интерфейса вместо кода C#.

XAML был разработан для упрощения процесса создания пользовательского интерфейса в приложениях.

Введение в XAML

.NET MAUI реализует синтаксический анализатор XAML, который анализирует объявленные вами элементы XAML и создает экземпляр каждого элемента как тип .NET.

Диалект XAML, который понимает средство синтаксического анализа .NET MAUI, специфичен для .NET MAUI, хотя он похож на XAML, используемый другими платформами, такими как Windows Presentation Foundation (WPF)

Введение в XAML

Большинство распространенных типов, используемых приложением MAUI, находятся в пакетах

Microsoft.Maui.Dependencies и
Microsoft.Maui.Extensions.

Введение в XAML

Большинство элементов управления MAUI расположены в пространстве имен **Microsoft.Maui.Controls**, в то время как пространство имен **Microsoft.Maui** определяет служебные типы, такие как **Thickness**, а пространство имен **Microsoft.Maui.Graphics** включает обобщенные типы, такие как **Color**.

Введение в XAML

По большей части вам не нужно беспокоиться об этих пространствах имен, поскольку они вводятся с помощью функции неявного использования C#, которая автоматически добавляет их во все приложения.

Основы разметки

ВВЕДЕНИЕ В XAML

Основы разметки

Для анализа XAML-определения элемента управления на странице синтаксический анализатор XAML должен иметь доступ к коду, который реализует элемент управления и определяет его свойства. Элементы управления, доступные для страницы .NET MAUI, реализованы в наборе сборок, которые устанавливаются как часть пакета Microsoft.Maui NuGet.

В коде C# вы вводите пространство имен в область видимости с помощью директивы **using**.

На странице XAML вы ссылаетесь на пространство имен с помощью атрибута **xmlns** страницы.

Основы разметки

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  
x:Class="MAUI_LK.MainPage">
```

...

```
</ContentPage>
```

Основы разметки

<http://schemas.microsoft.com/dotnet/2021/maui> является пространством имен по умолчанию для страницы.

Этот URI является просто псевдонимом для одного или нескольких пространств имен, определенных сборками в пакете Microsoft.Maui, поэтому указание этого пространства имен в начале страницы включает в себя все типы и элементы управления .NET MAUI.

Если вы опустите это пространство имен, вы не сможете использовать такие элементы управления, как Button, Label, Entry или StackLayout.

Основы разметки

<http://schemas.microsoft.com/winfx/2009/xaml>, ссылается на сборки, содержащие различные встроенные типы .NET, такие как строки, числовые значения и свойства.

В приведенном выше коде XAML этому пространству имен назначается псевдоним `x`. В коде XAML для этой страницы вы ссылаетесь на типы в этом пространстве имен, добавляя к ним префикс `x:`.

Например, каждая страница XAML компилируется в класс, и вы указываете имя класса, который создается с помощью атрибута `x:Class` страницы

Основы разметки

Для использования своих типов на странице вы можете ссылаться на них в коде XAML через пространство имен XAML. Например, если у вас есть типы и методы, которые вы хотите использовать в коде XAML, определенные в пространстве имен с именем Entities в вашем проекте, вы можете добавить пространство имен Entities на страницу

Основы разметки

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  
xmlns:entities="clr-namespace:MAUI_LK.Entities"  
x:Class="MAUI_LK.MainPage">
```

Основы разметки

Для описания или предоставления информации об элементе в XML вы используете атрибуты.

В XAML атрибуты используются для задания свойств базового типа.

Основы разметки

```
var label = new Label  
{ Text = "Username", TextColor = Colors.Black };
```

Этот оператор создает новый объект **Label** и устанавливает свойства **Text** и **TextColor**.

Соответствующий код XAML выглядит так:

```
<Label Text="Username" TextColor="Black" />
```


Основы разметки

Преобразователь типов используется для преобразования атрибута XML, указанного в виде строкового значения, в его правильный тип.

.NET MAUI имеет преобразователи типов для большинства встроенных классов, и он будет использовать эти преобразователи типов автоматически.

Основы разметки

Преобразователи типов отлично подходят для простых настроек свойств; однако в некоторых случаях необходимо создать полный объект с собственными значениями свойств.

Решение этой проблемы заключается в изменении назначения свойств для использования синтаксиса на основе элементов. Этот синтаксис называется формой **Property Element**.

Основы разметки

```
<Label Text="Username" TextColor="Black"  
      FontSize="42" FontAttributes="Bold,Italic">  
  <Label.GestureRecognizers>  
    <TapGestureRecognizer NumberOfTapsRequired="2" />  
  </Label.GestureRecognizers>  
</Label>
```

Компоновка представлений на странице

Компоновка представлений на странице

.NET MAUI предоставляет панели компоновки, помогающие создавать согласованные пользовательские интерфейсы.

Панель макета отвечает за изменение размера и расположение дочерних представлений.

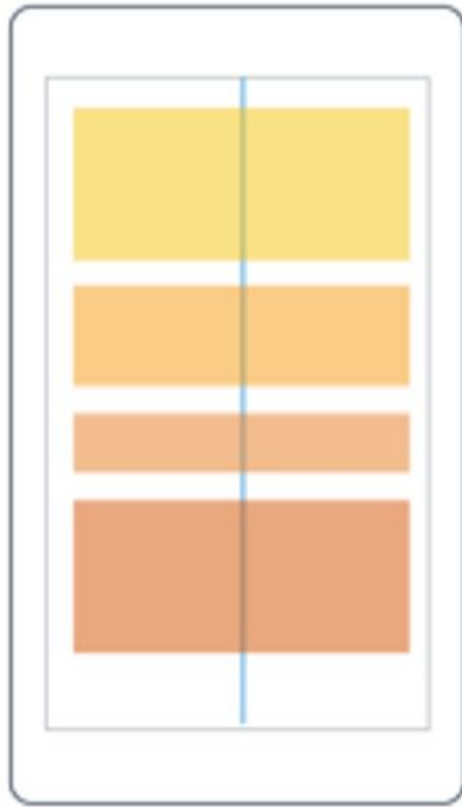
Компоновка представлений на странице

Layout panel — это контейнер .NET MAUI, который содержит коллекцию дочерних представлений и определяет их размер и положение. Layout panel автоматически пересчитывается при изменении размера приложения; например, когда пользователь поворачивает устройство.

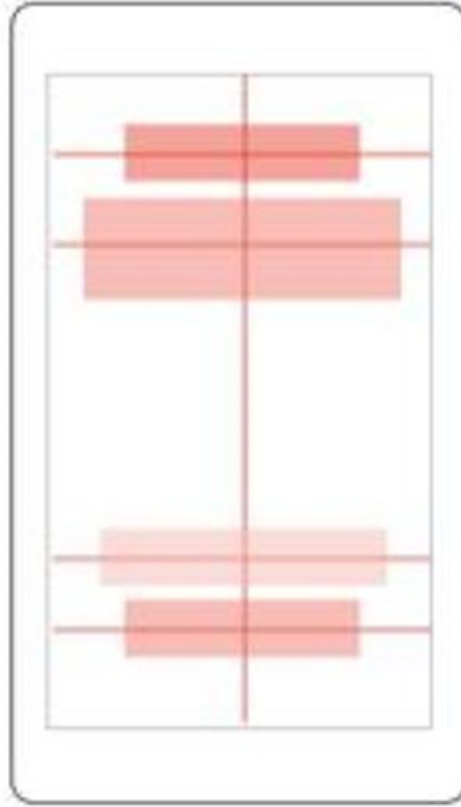
Примечание

Термин представление (View) или дочернее представление относится к элементу управления, размещенному на панели макета. Представлением может быть метка, кнопка, поле ввода или любой другой тип визуального элемента, поддерживаемый .NET MAUI.

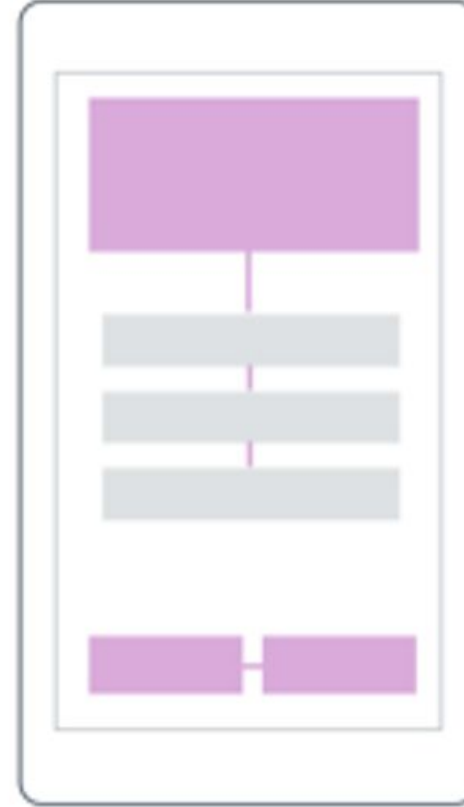
Компоновка представлений на странице



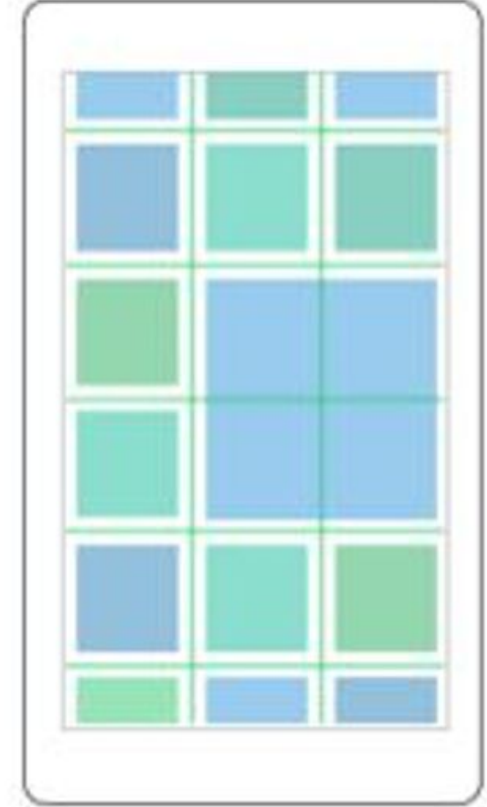
StackLayout



AbsoluteLayout



FlexLayout



Grid

Компоновка представлений на странице

StackLayout: размещает дочерние представления в одной строке или столбце.

В дополнение к StackLayout есть также новые оптимизированные **VerticalStackLayout** и **HorizontalStackLayout**, когда вам не нужно менять ориентацию.

Компоновка представлений на странице

AbsoluteLayout: размещает свое дочернее представление, используя координаты x и y .

Компоновка представлений на странице

FlexLayout: упорядочивает свои дочерние представления, как `StackLayout`, за исключением того, что вы можете переместить их (`wrap`), если они не помещаются в одну строку или столбец.

Компоновка представлений на странице

Grid: упорядочивает дочерние представления в ячейках, созданных строками и столбцами.

Компоновка представлений на странице

Grid: упорядочивает дочерние представления в ячейках, созданных строками и столбцами.

Компоновка представлений на странице

Базовый класс `View` определяет два свойства, которые влияют на размер представления: **`WidthRequest`** и **`HeightRequest`**. **`WidthRequest`** позволяет указать ширину, а **`HeightRequest`** позволяет указать высоту. Оба свойства имеют тип **`double`**.

Компоновка представлений на странице

<Label

Text="Hello"

BackgroundColor="Silver"

VerticalOptions="Center"

HorizontalOptions="Center"

WidthRequest="100"

HeightRequest="300"

FontSize="40"/>

Компоновка представлений на странице

Значения **WidthRequest** и **HeightRequest**, не имеют единиц измерения. Это не точки или пиксели. Это просто значения типа **double**. .NET MAUI передает эти значения базовой операционной системе во время выполнения. Это операционная система, которая предоставляет контекст, необходимый для определения того, что означают числа:

- ❑ В iOS значения называются точками (*points*).
- ❑ В Android это пиксели, не зависящие от плотности (*density-independent pixels*).

Компоновка представлений на странице

Базовый класс **View** имеет два свойства, которые используются для установки положения представления:

- ❑ **VerticalOptions** и
- ❑ **HorizontalOptions**.

Эти настройки влияют на то, как представление позиционируется в пределах прямоугольника, выделенного для него панелью макета.

VerticalOptions и **HorizontalOptions** относятся к типу **LayoutOptions**.

Компоновка представлений на странице

LayoutOptions — это тип C#, который инкапсулирует два параметра макета: **LayoutAlignment** и **Expands**. Оба свойства связаны с позиционированием, но не связаны друг с другом.

```
C#  
  
public struct LayoutOptions  
{  
    public LayoutAlignment Alignment { get; set; }  
    public bool Expands { get; set; }  
    ...  
}
```

Компоновка представлений на странице

LayoutAlignment — это перечисление, содержащее четыре значения: **Start**, **Center**, **End** и **Fill**. Вы можете использовать эти значения для управления тем, как дочернее представление позиционируется в прямоугольнике, заданном ему его панелью макета.

Компоновка представлений на странице

<StackLayout>

```
<Label Text="Start" HorizontalOptions="Start"
      BackgroundColor="Silver" FontSize="40" />
```

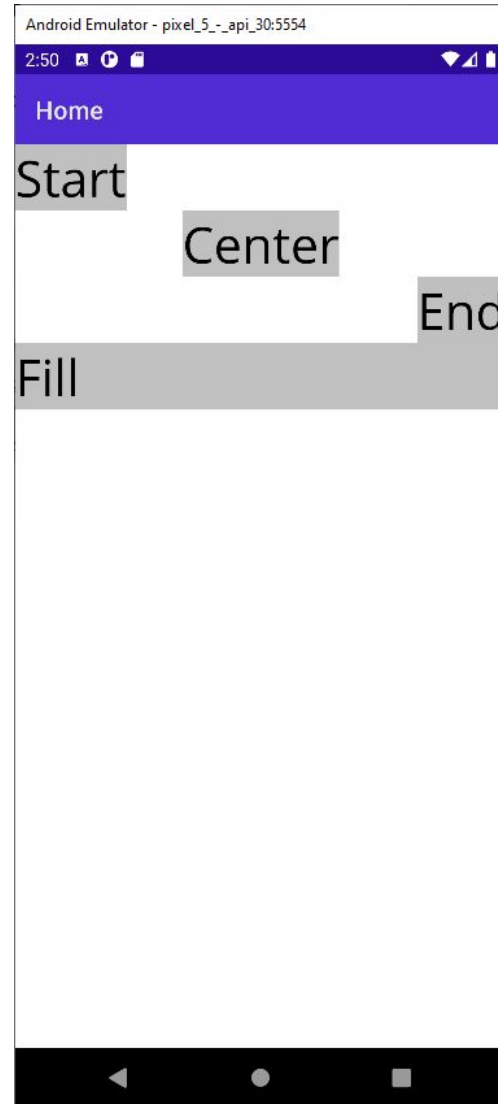
```
<Label Text="Center" HorizontalOptions="Center"
      BackgroundColor="Silver" FontSize="40" />
```

```
<Label Text="End" HorizontalOptions="End"
      BackgroundColor="Silver" FontSize="40"/>
```

```
<Label Text="Fill" HorizontalOptions="Fill"
      BackgroundColor="Silver" FontSize="40"/>
```

</StackLayout>

Компоновка представлений на странице



Элементы интерфейса (Views)

Представление данных

- ☐ Border
- ☐ Frame
- ☐ GraphicsView
- ☐ Image
- ☐ Label
- ☐ ScrollView
- ☐ WebView

Представление данных

Border — это контейнерный элемент управления, который рисует границу, фон или и то, и другое вокруг другого элемента управления.

Border может содержать только один дочерний объект.

Представление данных

```
<Border Stroke="#C49B33" StrokeThickness="4"  
    StrokeShape="RoundRectangle 40,0,0,40"  
    Background="#2B0B98" Padding="16,8"  
    HorizontalOptions="Center">  
    <Label Text=".NET MAUI"  
        TextColor="White"  
        FontSize="18"  
        FontAttributes="Bold" />  
</Border>
```

ИЛИ

```
<Border Stroke="#C49B33" StrokeThickness="4"  
    Background="#2B0B98" Padding="16,8"  
    HorizontalOptions="Center">  
    <Border.StrokeShape>  
        <RoundRectangle CornerRadius="40,0,0,40" />  
    </Border.StrokeShape>  
    <Label Text=".NET MAUI"  
        TextColor="White"  
        FontSize="18"  
        FontAttributes="Bold" />  
</Border>
```



Представление данных

Image отображает изображение, которое можно загрузить из локального файла, URI, встроенного ресурса или потока. Поддерживаются стандартные форматы изображений платформы, включая анимированные GIF, а также поддерживаются локальные файлы масштабируемой векторной графики (SVG).

Представление данных

<Image

Source="dotnet_bot.png"

HeightRequest="200"

HorizontalOptions="Center" />

Image image = new Image

{

Source = ImageSource.FromFile("dotnet_bot.png")

};

Представление данных

<Image

Source="https://aka.ms/campus.jpg"

HeightRequest="200"

HorizontalOptions="Center" />

Image image = new Image

{

Source = ImageSource.FromUri("https://aka.ms/campus.jpg")

};

Представление данных

Label отображает однострочный и многострочный текст. Текст, отображаемый меткой, может быть цветным, разделенным и может иметь текстовые украшения.

```
<Label Text=".NET MAUI"  
  TextColor="White"  
  FontSize="18"  
  FontAttributes="Bold" />
```

Представление данных

ScrollView — это представление, способное прокручивать содержимое.

По умолчанию ScrollView прокручивает содержимое вертикально.

ScrollView может иметь только один дочерний элемент, хотя это могут быть и другие макеты.

Представление данных

WebView отображает удаленные веб-страницы, локальные файлы HTML и строки HTML в приложении. Содержимое, отображаемое **WebView**, включает поддержку каскадных таблиц стилей (CSS) и JavaScript.

По умолчанию проекты .NET MAUI включают разрешения платформы, необходимые для **WebView** для отображения удаленной веб-страницы.

Представление данных

GraphicsView

Генерирование команд

Генерирование команд

☐ Button

☐ ImageButton

☐ RadioButton

☐ RefreshView

☐ SearchBar

☐ SwipeView

Генерирование команд

Button отображает текст и реагирует на прикосновение или щелчок, который направляет приложение на выполнение задачи.

Button обычно отображает короткую текстовую строку, указывающую на команду, но также может отображать растровое изображение или комбинацию текста и изображения. Когда кнопка нажимается пальцем или щелкается мышью, она инициирует эту команду.

Генерирование команд

```
<Button  
  x:Name="CounterBtn"  
  Text="Click me"  
  Clicked="OnCounterClicked"  
  HorizontalOptions="Center" />
```

```
private void OnCounterClicked(object sender, EventArgs e)  
{  
    count++;  
    if (count == 1)  
        CounterBtn.Text = $"Clicked {count} time";  
    else  
        CounterBtn.Text = $"Clicked {count} times";  
}
```

Генерирование команд

Button отображает текст и реагирует на прикосновение или щелчок, который направляет приложение на выполнение задачи.

Button обычно отображает короткую текстовую строку, указывающую на команду, но также может отображать растровое изображение или комбинацию текста и изображения. Когда кнопка нажимается пальцем или щелкается мышью, она инициирует эту команду.

Установка значений

Установка значений

- ☐ CheckBox
- ☐ DatePicker
- ☐ Slider
- ☐ Stepper
- ☐ Switch
- ☐ TimePicker

Установка значений

CheckBox — это тип кнопки, которая может быть либо отмечена, либо пуста. Когда флажок установлен, он считается включенным. Когда флажок пуст, он считается выключенным.

Редактирование текста

Редактирование текста

 Editor

 Entry

Редактирование текста

Editor позволяет редактировать многострочный текст

```
<Editor x:Name="editor"  
    Placeholder="Enter your response here"  
    HeightRequest="250"  
    TextChanged="OnEditorTextChanged"  
    Completed="OnEditorCompleted" />
```

Редактирование текста

Entry позволяет вводить и редактировать одну строку текста.

Кроме того, **Entry** можно использовать в качестве поля пароля.

Обработка событий

Обработка событий

```
<Button  
    x:Name="CounterBtn"  
    Text="Click me"  
    Clicked="OnCounterClicked"  
    HorizontalOptions="Center" />
```

```
private void OnCounterClicked(object sender, EventArgs e)  
{  
    count++;  
    if (count == 1)  
        CounterBtn.Text = $"Clicked {count} time";  
    else  
        CounterBtn.Text = $"Clicked {count} times";  
}
```

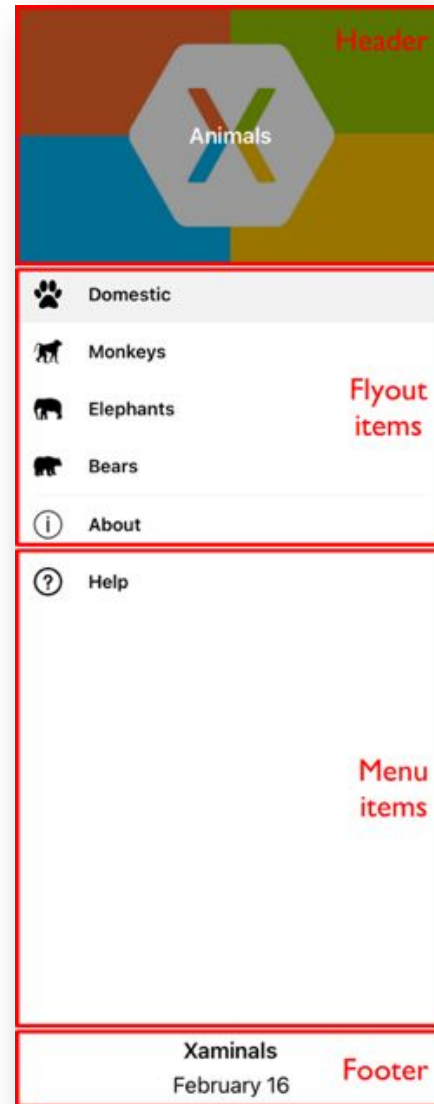
Навигация страниц

Навигация страниц

Всплывающая навигация (**Flyout**) отображает меню, которое обеспечивает быстрый способ переключения контекста в вашем приложении.

Всплывающее меню состоит из нескольких частей: заголовка, всплывающих элементов, элементов меню и нижнего колонтитула.

Навигация страниц



Навигация страниц

Для реализации всплывающей навигации в .NET MAUI. используется класс **FlyoutItem**.

Вы указываете, что будет отображаться при касании **FlyoutItem**, устанавливая его свойство **ShellContent**. Это свойство будет указывать на страницу в приложении.

FlyoutItem должен размещаться на странице **Shell**, которая служит главной страницей приложения.

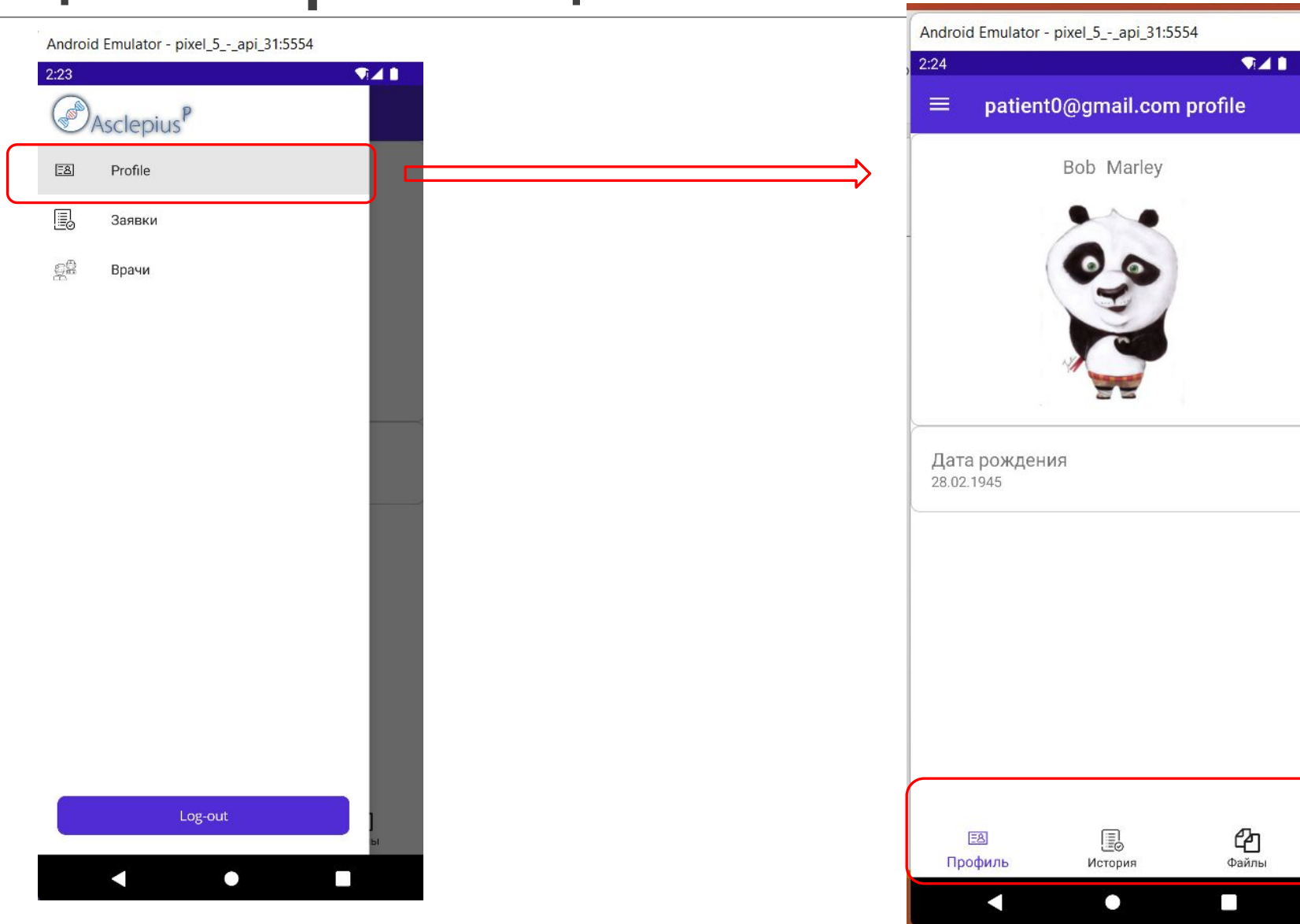
Навигация страниц

Объект **Shell** может содержать только объекты **FlyoutItem** или объект **TabBar** (который может содержать только объекты **Tab**), которые могут содержать только объекты **ShellContent**.

Навигация страниц

```
<Shell.FlyoutHeader>
  <VerticalStackLayout Padding="5">
    <Image Source="asclp_rct.svg" HorizontalOptions="Start" HeightRequest="50">
      <Image.Shadow>
        <Shadow/>
      </Image.Shadow>
    </Image>
  </VerticalStackLayout>
</Shell.FlyoutHeader>
<FlyoutItem Title="Profile" Route="CardProfile" Icon="profile.svg">
  <ShellContent Title="Профиль" Icon="profile.svg" ContentTemplate="{DataTemplate card:CardProfile}"/>
  ...
  <ShellContent Title="Файлы" Icon="files.svg" ContentTemplate="{DataTemplate card:Files}" />
</FlyoutItem>
<FlyoutItem Title="Заявки" Route="OrdersList" Icon="list.svg">
  <ShellContent Title="История" Icon="list.svg" ContentTemplate="{DataTemplate order:OrdersList}" />
</FlyoutItem>
...
<Shell.FlyoutFooter>
  <VerticalStackLayout Padding="20">
    <Button Text="Log-out" Clicked="Button_Clicked"></Button>
  </VerticalStackLayout>
</Shell.FlyoutFooter>
```

Навигация страниц



Навигация по стеку

Навигация по стеку позволяет пользователю развернуть ряд страниц, где следующая страница в стеке обеспечивает более подробное представление выбранного элемента на предыдущей странице.

Навигация по стеку

Shell включает в себя навигацию на основе **URI**, которая использует маршруты для перехода на любую страницу в приложении без необходимости следовать установленной иерархии навигации. Кроме того, он также предоставляет возможность навигации назад без необходимости посещения всех страниц в стеке навигации.

Навигация по стеку

Класс `Shell` определяет следующие свойства, связанные с навигацией:

`BackButtonBehavior`, тип `BackButtonBehavior` — присоединенное свойство, определяющее поведение кнопки «Назад».

`CurrentItem`, типа `ShellItem`, текущий выбранный элемент.

`CurrentPage`, тип `Page`, текущая представленная страница.

`CurrentState`, тип `ShellNavigationState`, текущее состояние навигации оболочки.

`Current`, тип `Shell`, псевдоним `Application.Current.MainPage` с приведенным типом.

Навигация по стеку

Навигация выполняется путем вызова метода

GoToAsync

из класса **Shell**

Навигация по стеку

Маршруты могут быть определены для объектов **FlyoutItem**, **TabBar**, **Tab** и **ShellContent** через их свойства **Route**:

```
<FlyoutItem FlyoutItemsVisible="False" Shell.FlyoutBehavior="Disabled">  
    <ShellContent ContentTemplate="{DataTemplate pages:StartupPage}"  
        Route="StartupPage"/>  
</FlyoutItem>
```

Навигация по стеку

В конструкторе подкласса **Shell** дополнительные маршруты могут быть явно зарегистрированы для любых **ContentPage**, которые не представлены в визуальной иерархии **Shell**. Это достигается с помощью метода **Routing.RegisterRoute**:

C#

```
Routing.RegisterRoute("astronomicalbodydetails",  
    typeof(AstronomicalBodyPage));
```

Навигация по стеку

Чтобы перейти к **AstronomicalBodyPage**, нужно вызвать:

C#

```
await Shell.Current.GoToAsync("astronomicalbodydetails");
```

Навигация по стеку

Примитивные данные могут быть переданы как строковые параметры запроса при выполнении программной навигации на основе URI. Это достигается добавлением ? после маршрута, за которым следует идентификатор параметра запроса, = и значение:

```
string celestialName = "moon";
```

```
await
```

```
Shell.Current.GoToAsync($"astronomicalbodydetails?bodyName={celestialName}");
```

Навигация по стеку


Передача данных в виде параметров:

```
IDictionary<string, object> parameters =  
    new Dictionary<string, object>  
    {  
        {"Doctor", doctorDetails}  
    };  
  
await Shell.Current.GoToAsync("DoctorDetails", parameters);
```

Навигация по стеку

Получение параметров:

```
[QueryProperty(nameof(DoctorInfo), "Doctor")]  
public partial class DoctorDetailsViewModel : ViewModelBase  
{  
    public Doctor DoctorInfo {get;set;};  
  
    ...  
}
```



Навигация по стеку

Поддерживаются следующие форматы относительных маршрутов.

route	Поиск указанного маршрута выполняется в иерархии маршрутов вверх, начиная от текущей позиции. Страница совпадения будет отправлена в стек навигации.
/маршрут	Поиск указанного маршрута выполняется в иерархии маршрутов вниз, начиная от текущей позиции. Страница совпадения будет отправлена в стек навигации.
//маршрут	Поиск указанного маршрута выполняется в иерархии маршрутов вверх, начиная от текущей позиции. Страница совпадения заменит стек навигации .
///маршрут	Поиск указанного маршрута выполняется в иерархии маршрутов вниз, начиная от текущей позиции. Страница совпадения заменит стек навигации .