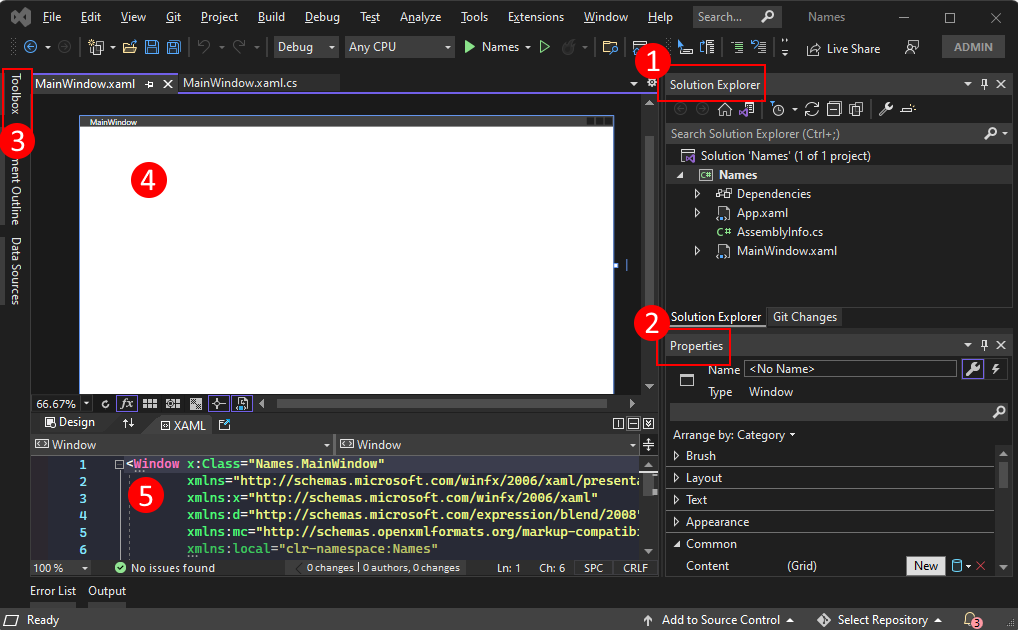
### Важные элементы среды Visual Studio

Поддержка WPF в Visual Studio состоит из пяти важных компонентов, с которыми вы будете взаимодействовать при создании приложения.



1. Обозреватель решений

Все файлы проекта, код, окна и ресурсы отображаются в этой области.

1. Properties (Свойства)

На этой панели отображаются параметры свойств, которые можно настроить в зависимости от выбранного элемента. Например, если выбрать элемент в **Обозревателе решений**, отобразятся параметры свойств, связанные с файлом. Если выбрать объект в **конструкторе**, отобразятся параметры этого элемента.

1. Панель элементов

Панель элементов содержит все элементы управления, которые можно добавить на форму. Чтобы добавить элемент управления на текущую форму, дважды щелкните элемент управления или перетащите его.

1. Конструктор XAML

Это конструктор для документа XAML. Он является интерактивным, и на него можно перетаскивать объекты из **панели элементов**. Выбирая и перемещая элементы в конструкторе, можно визуально создавать пользовательский интерфейс для приложения.

Если конструктор и редактор отображаются, изменения в одном из них сразу отражаются в другом. При выборе элементов в конструкторе в области **Свойства** отображаются свойства и атрибуты этого объекта.

1. Редактор кода XAML

Это редактор кода XAML для документа XAML. Редактор кода XAML — это способ создания пользовательского интерфейса вручную без конструктора. Конструктор может вычислять значения свойств элемента управления при его добавлении в конструктор. В редакторе кода XAML вам предоставляется гораздо больше контроля.

Если конструктор и редактор отображаются, изменения в одном из них сразу отражаются в другом. При переходе по текстовым курсорам в редакторе кода в области **Свойства** отображаются свойства и атрибуты этого объекта.

## Изучение кода XAML

После создания проекта в редакторе кода XAML отображается минимальный объем кода XAML для отображения окна. Если редактор не открыт, дважды щелкните элемент MainWindow.xaml в **обозревателе решений**. Вы должны увидеть XAML, аналогичный приведенному ниже примеру.

XAML

<Window x:Class="Names.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:Names"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

</Grid>

</Window>

Давайте развернем этот код XAML, чтобы лучше понять его. XAML — это просто XML, который может обрабатываться компиляторами, используемыми в WPF. Он описывает пользовательский интерфейс WPF и взаимодействует с кодом .NET. Для понимания XAML следует, как минимум, знать основы XML.

Корень документа <Window> представляет тип объекта, описываемого файлом XAML. Объявлено восемь атрибутов, и обычно они относятся к трем категориям:

* Пространства имен

Пространство имен XML предоставляет структуру XML, определяя, какое содержимое XML может быть объявлено в файле.

Основной атрибут xmlns импортирует пространство имен XML для всего файла, а в данном случае сопоставляется с типами, объявленными в WPF. Другие пространства имен XML объявляют префикс и импортируют другие типы и объекты для XAML-файла. Например, пространство имен xmlns:local объявляет префикс local и выполняет сопоставление объектов, объявленных в проекте, с теми, которые объявлены в пространстве имен кода Names.

* Атрибут x:Class

Этот атрибут сопоставляет <Window> с типом, определенным в коде: файл MainWindow.xaml.cs или MainWindow.xaml.vb, который является классом Names.MainWindow.

* Атрибут Title

Любой обычный атрибут, объявленный в объекте XAML, задает свойство этого объекта. В этом случае Title атрибут задает Window.Title свойство.

## Изменение окна

Сначала запустите проект и просмотрите выходные данные по умолчанию. Вы увидите окно, которое появляется, без элементов управления, и заголовок **MainWindow**:



Для нашего примера приложения это окно слишком велико, а его заголовок не является описательным. Измените заголовок и размер окна, заменив соответствующие атрибуты в XAML следующими значениями:

XAMLКопировать

<Window x:Class="Names.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:Names"

mc:Ignorable="d"

Title="Names" Height="180" Width="260">

<Grid>

</Grid>

</Window>

## Подготовка макета

В WPF имеется система макетов с широкими возможностями с множеством различных элементов управления макетом. Элементы управления макета помогают размещать дочерние элементы управления и изменять их размеры, а также могут даже это делать автоматически. Элемент управления макета по умолчанию, предоставляемый вам в этом XAML, — это элемент управления <Grid>.

Элемент управления Grid позволяет определить строки и столбцы, почти как в таблице, и размещать элементы управления внутри границ определенной комбинации строк и столбцов. Можно добавить любое количество дочерних элементов управления или других элементов управления макетом, имеющихся в Grid. Например, другой элемент управления Grid можно поместить в определенную комбинацию строк и столбцов, а для этого нового Grid затем можно задать больше строк и столбцов и собственные дочерние элементы.

Элемент управления <Grid> определяет строки и столбцы, в которых будут находиться элементы управления. В сетке всегда объявлена одна строка и столбец, то есть сетка по умолчанию является отдельной ячейкой. Это не обеспечивает большую гибкость при размещении элементов управления.

Прежде чем добавлять новые строки и столбцы, добавьте новый атрибут в элемент <Grid> : Margin="10". Будет вставлена сетка, благодаря чему окно будет выглядеть немного лучше.

Затем определите две строки и два столбца, разделив сетку на четыре ячейки:

XAML

<Window x:Class="Names.LayoutStep2"

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:Names"

mc:Ignorable="d"

Title="Names" Height="180" Width="260">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

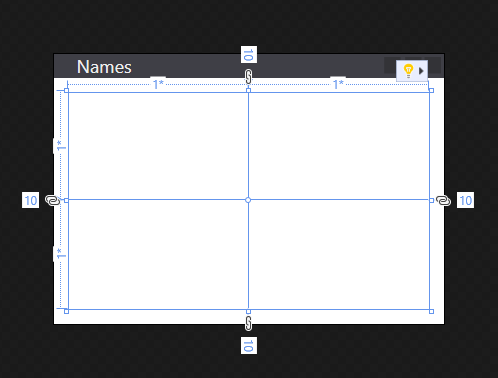
<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

</Grid>

</Window>

Выберите сетку в редакторе кода XAML или в конструкторе XAML. Вы увидите, что в конструкторе XAML отображается каждая строка и столбец:



## Добавление первого элемента управления

После создания сетки можно приступать к добавлению в нее элементов управления. Начните с элемента управления Label. Создайте новый элемент <Label> внутри элемента <Grid> после определений строк и столбцов, а затем присвойте ему строковое значение Names:

XAMLКопировать

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Label>Names</Label>

</Grid>

<Label>Names</Label> определяет содержимое Names. Некоторым элементам управления известно, как управлять содержимым, а другим — нет. Содержимое элемента управления сопоставляется со свойством Content. При задании содержимого с помощью синтаксиса атрибутов XAML можно использовать следующий формат: <Label Content="Names" />. Оба способа служат для выполнения одного и того же действия — отображение текста Names в качестве содержимого метки.

У нас есть проблема, поскольку метка занимает половину окна, так как она была автоматически назначена первой строке и столбцу сетки. Для первой строки нам не нужно много пространства, поскольку мы будем использовать эту строку только для метки. Измените атрибут Height первого параметра <RowDefinition> с \* на Auto. Значение Auto автоматически изменяет размер строки сетки под размер ее содержимого (в данном случае это элемент управления Label).

XAMLКопировать

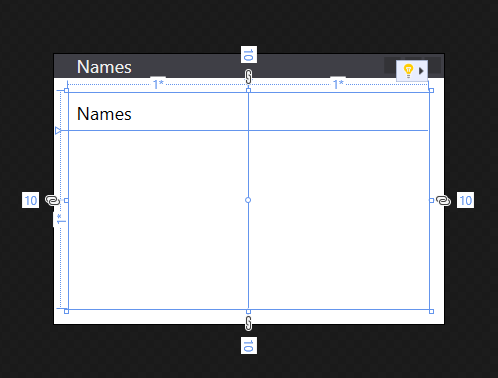
<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

Обратите внимание, что теперь в конструкторе отображается метка, занимающая небольшую часть доступной высоты. Теперь имеется больше места для следующей строки. Большинство элементов управления определяют значения высоты и ширины, которые они должны занимать. Например, элемент управления "метка" имеет значение высоты, которое гарантирует, что его можно прочитать.



### Размещение элементов управления

Давайте поговорим о размещении элементов управления. Метка, созданная в разделе выше, автоматически помещается в строку 0 и столбец 0 сетки. Нумерация строк и столбцов начинается с 0 и увеличивается на 1 для каждой новой строки или столбца. Элементу управления ничего не известно о сетке, и элемент управления не определяет свойства для управления своим размещением в сетке. Элемент управления может быть даже помещен в другой элемент управления макета, имеющий собственный набор правил, определяющих способ размещения элементов управления.

Как указать элементу управления, что ему следует использовать другую строку или столбец, если элемент управления не имеет сведений о сетке? В этом случае на помощь придут вложенные свойства! Сетка использует эффективную систему свойств, предоставляемую WPF. Сетка определяет новые свойства, которые дочерние элементы управления могут объявлять и использовать. Свойства фактически не существуют в самом элементе управления, они вкладываются сеткой при добавлении элемента управления в сетку.

Сетка определяет два свойства для определения размещения строк и столбцов дочернего элемента управления: Grid.Row и Grid.Column. Если эти свойства опущены в элементе управления, предполагается, что они имеют значения по умолчанию 0, поэтому элемент управления помещается в строку 0 и столбец 0 сетки. Попробуйте изменить расположение элемента управления <Label>, присвоив атрибуту Grid.Column значение 1:

XAMLКопировать

<Label Grid.Column="1">Names</Label>

Обратите внимание, что метка теперь переместилась во второй столбец. Вложенные свойства Grid.Row и Grid.Column можно использовать для размещения последующих элементов управления, которые мы создадим. Но на этом этапе следует вернуть метку в строку 0.

## Создание поля со списком имен

Теперь, когда сетка имеет правильный размер и создана метка, добавьте элемент управления "Список" в строку под меткой. Список будет находиться в строке 1 и столбце 0. Присвоим этому элементу управления имя lstNames. После именования элемента управления на него можно ссылаться в коде программной части. Имя присваивается элементу управления с помощью атрибута x:Name.

XAML

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Label>Names</Label>

<ListBox Grid.Row="1" x:Name="lstNames" />

</Grid>

## Добавление оставшихся элементов управления

Последние два элемента управления, которые предстоит добавить, — это текстовое поле и кнопка, которые пользователь будет использовать для ввода имени, добавляемого в список. Однако вместо того, чтобы пытаться создать больше строк и столбцов для сетки, эти элементы управления будут размещены в элементе управления <StackPanel>.

Панель стека отличается от сетки тем, как в ней размещаются элементы управления. В сетке для указания того, где должны располагаться элементы управления, вы используете свойства Grid.Row и Grid.Column, тогда как в панели стека это выполняется автоматически: сначала размещается первый элемент управления, затем после него следующий элемент управления, и так до тех пор, пока не будут размещены все элементы управления. Элементы управления размещаются в "стек" один под другим.

Создайте элемент управления <StackPanel> после списка и вставьте его в строку 1 и столбец 1 сетки. Добавьте еще один атрибут с именем Margin и значением 5,0,0,0:

XAML

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Label>Names</Label>

<ListBox Grid.Row="1" x:Name="lstNames" />

<StackPanel Grid.Row="1" Grid.Column="1" Margin="5,0,0,0">

</StackPanel>

Атрибут Margin ранее уже использовался в сетке, однако теперь мы поместим только одно значение, 10. Мы использовали значение 5,0,0,0 в панели стека. Поле является типом Thickness и может интерпретировать оба значения. Толщина определяет пространство вокруг каждой стороны прямоугольника, **слева**, **сверху**, **справа**, **снизу**, соответственно. Если значение поля является одним значением, оно используется для всех четырех сторон.

Затем создайте <TextBox> и элемент управления <Button> в <StackPanel>.

XAML

<StackPanel Grid.Row="1" Grid.Column="1" Margin="5,0,0,0">

<TextBox x:Name="txtName" />

<Button x:Name="btnAdd" Margin="0,5,0,0">Add Name</Button>

</StackPanel>

Макет окна готов. Однако в наше приложение следует добавить логику, чтобы оно работало. Далее необходимо подключить события элемента управления к коду, чтобы приложение выполняло действия.

## Добавление кода для события Click

Созданная нами кнопка <Button> имеет событие Click, возникающее, когда пользователь нажимает кнопку. Можно подписаться на это событие и добавить код, который будет добавлять имя в список. Как и при задании свойства элемента управления путем добавления атрибута XAML, можно использовать атрибут XAML для подписки на событие. Задайте для атрибута Click значение ButtonAddName\_Click.

XAMLь

<StackPanel Grid.Row="1" Grid.Column="1" Margin="5,0,0,0">

<TextBox x:Name="txtName" />

<Button x:Name="btnAdd" Margin="0,5,0,0" Click="ButtonAddName\_Click">Add Name</Button>

</StackPanel>

Теперь необходимо создать код обработчика. Щелкните правой кнопкой мыши ButtonAddName\_Click и выберите **Перейдите к определению**. Это действие создает метод в коде программной части, который соответствует введенному имени обработчика.

C#

private void ButtonAddName\_Click(object sender, RoutedEventArgs e)

{

}

Затем добавьте следующий код для выполнения следующих трех шагов:

1. Проверка того, что в текстовом поле содержится имя.
2. Проверка того, что имя, указанное в текстовом поле, еще не добавлено.
3. Добавление имени в список.

C#

private void ButtonAddName\_Click(object sender, RoutedEventArgs e)

{

if (!string.IsNullOrWhiteSpace(txtName.Text) && !lstNames.Items.Contains(txtName.Text))

{

lstNames.Items.Add(txtName.Text);

txtName.Clear();

}

}

## Запустите приложение

Теперь, когда у нас есть код события, можно запустить приложение, нажав клавишу F5 или выбрав пункт меню **Отладка**>**Начать отладку**. Отобразится окно, и вы можете ввести имя в текстовое поле, а затем добавить его, нажав кнопку.

