В этом руководстве вы узнаете, как:

* Создайте проект WPF.
* Используйте XAML для проектирования внешнего вида пользовательского интерфейса приложения.
* Напишите код для создания поведения приложения.
* Создайте определение приложения для управления приложением.
* Добавьте элементы управления и создайте макет, чтобы составить пользовательский интерфейс приложения.
* Создание стилей для согласованного внешнего вида в пользовательском интерфейсе приложения.
* Привязка пользовательского интерфейса к данным для заполнения пользовательского интерфейса данными и синхронизации данных и пользовательского интерфейса.

по завершении работы с этим руководством вы создадите автономное приложение Windows, которое позволяет пользователям просматривать отчеты о расходах для выбранных лиц. Приложение состоит из нескольких страниц WPF, размещенных в окне в стиле браузера.

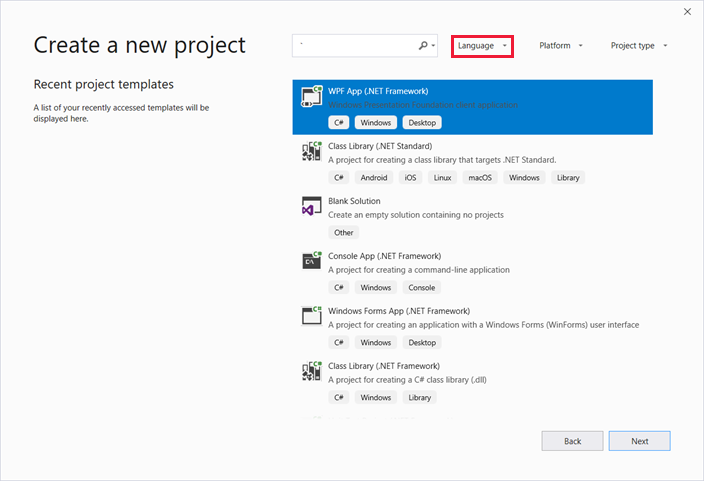
**Создание проекта приложения**

Первым шагом является создание инфраструктуры приложения, включающей определение приложения, две страницы и изображение.

1. создайте новый проект приложения WPF в Visual Basic или Visual C# с именем **ExpenseIt** :
   1. откройте Visual Studio и выберите **создать новый проект** в меню **начало работы** .

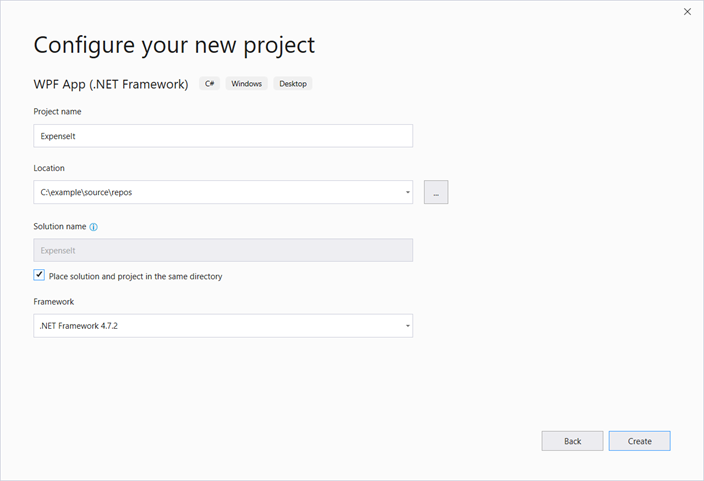
Откроется диалоговое окно **Создание нового проекта** .

* 1. В раскрывающемся списке **язык** выберите либо **C#** , либо **Visual Basic**.
  2. выберите шаблон **приложения WPF (платформа .NET Framework)** и нажмите кнопку **далее**.



Откроется диалоговое окно **Настройка нового проекта** .

* 1. Введите имя **ExpenseIt** проекта и нажмите кнопку **создать**.



Visual Studio создает проект и открывает конструктор для окна приложения по умолчанию с именем **MainWindow. xaml**.

1. откройте *App. xaml* (C#).

Этот XAML-файл определяет приложение WPF и все ресурсы приложения. Этот файл также используется для указания пользовательского интерфейса, в данном случае *MainWindow. XAML*, который автоматически отображается при запуске приложения.

КОД XAML должен выглядеть следующим образом в C#:

XAML

<Application x:Class="ExpenseIt.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

StartupUri="MainWindow.xaml">

<Application.Resources>

</Application.Resources>

</Application>

1. Откройте файл *MainWindow.xaml*.

Этот XAML-файл является главным окном приложения и отображает содержимое, созданное на страницах. Класс определяет свойства окна, такие как заголовок, размер или значок, а также обрабатывает события, такие как закрытие или скрытие.

1. Измените элемент на NavigationWindow , как показано в следующем коде XAML:

XAML

<NavigationWindow x:Class="ExpenseIt.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

...

</NavigationWindow>

Это приложение переходит к другому содержимому в зависимости от введенных пользователем данных. Именно поэтому необходимо изменить основное Window значение на NavigationWindow . NavigationWindow наследует все свойства Window . NavigationWindow Элемент в файле XAML создает экземпляр NavigationWindow класса.

Grid Удалите элементы из между NavigationWindow тегами.

1. Измените следующие свойства в коде XAML для NavigationW
   1. Title Присвойте свойству значение " ExpenseIt ".
   2. Height Задайте для свойства значение 350 пикселей.
   3. Width Задайте для свойства значение 500 пикселей.

C#:

XAML

<NavigationWindow x:Class="ExpenseIt.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="ExpenseIt" Height="350" Width="500">

</NavigationWindow>

1. Откройте *MainWindow. XAML. CS*.

Этот файл является файлом кода программной части, который содержит код для работы с событиями, объявленными в файле *MainWindow. XAML*. Этот файл содержит разделяемый класс для окна, определенного в XAML-коде.

1. Если вы используете C#, измените MainWindow класс на производный от NavigationWindow Код C# теперь должен выглядеть следующим образом:

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ExpenseIt

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : NavigationWindow

{

public MainWindow()

{

InitializeComponent();

}

}

}

**Добавление файлов в приложение**

В этом разделе вы добавите в приложение две страницы и изображение.

1. Добавьте в проект новую страницу и назовите ее *ExpenseItHome.xaml* :
   1. в **Обозреватель решений** щелкните правой кнопкой мыши **ExpenseIt** узел проекта и выберите команду **добавить**>**страницу**.
   2. В диалоговом окне **Добавление нового элемента** шаблон **Page (WPF)** уже выбран. Введите имя **ExpenseItHome** и нажмите кнопку **Добавить**.

Эта страница является первой страницей, отображаемой при запуске приложения. Отобразится список людей для выбора, чтобы отобразить отчет о расходах для.

1. Откройте *ExpenseItHome.xaml*.
2. Title Присвойте свойству значение " ExpenseIt - Home ".
3. Задайте для параметра значение DesignHeight 350 пикселей, а для параметра DesignWidth — значение 500 пикселей.

C#:

<Page x:Class="ExpenseIt.ExpenseItHome"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

d:DesignHeight="350" d:DesignWidth="500"

Title="ExpenseIt - Home">

<Grid>

</Grid>

</Page>

1. Откройте файл *MainWindow.xaml*.
2. Source Добавьте свойство в NavigationWindow элемент и задайте для него значение " ExpenseItHome.xaml ".

Этот набор устанавливается *ExpenseItHome.xaml* в качестве первой страницы, открытой при запуске приложения.

C#:

XAML

<NavigationWindow x:Class="ExpenseIt.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

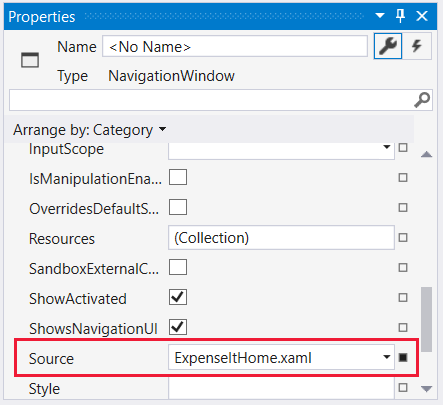
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="ExpenseIt" Height="350" Width="500" Source="ExpenseItHome.xaml">

</NavigationWindow>

**Совет**

Также можно задать свойство **Source** в категории **Разное** в окне **Свойства** .



1. Добавьте еще одну новую страницу WPF в проект и назовите ее *файл ExpenseReportPage. XAML*::
   1. в **Обозреватель решений** щелкните правой кнопкой мыши **ExpenseIt** узел проекта и выберите команду **добавить**>**страницу**.
   2. В диалоговом окне **Добавление нового элемента** выберите шаблон **Page (WPF)** . Введите имя **файл ExpenseReportPage** и нажмите кнопку **Добавить**.

На этой странице отображается отчет о расходах для пользователя, выбранного на **ExpenseItHome** странице.

1. Откройте *файл ExpenseReportPage. XAML*.
2. Title Присвойте свойству значение " ExpenseIt - View Expense ".
3. Задайте для параметра значение DesignHeight 350 пикселей, а для параметра DesignWidth — значение 500 пикселей.

C#:

<Page x:Class="ExpenseIt.ExpenseReportPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

d:DesignHeight="350" d:DesignWidth="500"

Title="ExpenseIt - View Expense">

<Grid>

</Grid>

</Page>

1. Откройте  *ExpenseItHome. XAML. CS* и *файл ExpenseReportPage. XAML. CS*.

при создании нового файла подкачки Visual Studio автоматически создает файл *кода программной части* . Эти файлы кода программной части обрабатывают логику, реагирующую на действия пользователя.

Код должен выглядеть следующим **ExpenseItHome** образом:

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ExpenseIt

{

/// <summary>

/// Interaction logic for ExpenseItHome.xaml

/// </summary>

public partial class ExpenseItHome : Page

{

public ExpenseItHome()

{

InitializeComponent();

}

}

}

И, как и для **файл ExpenseReportPage**:

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ExpenseIt

{

/// <summary>

/// Interaction logic for ExpenseReportPage.xaml

/// </summary>

public partial class ExpenseReportPage : Page

{

public ExpenseReportPage()

{

InitializeComponent();

}

}

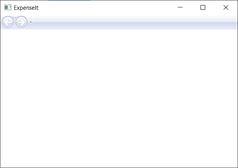
}

1. Добавьте в проект образ с именем *watermark.png* . вы можете создать собственный образ, скопировать файл из примера кода или получить его из репозитория microsoft/WPF-samples GitHub.
   1. Щелкните правой кнопкой мыши узел проекта и выберите команду **Добавить**>**существующий элемент** или нажмите клавиши **SHIFT**+**ALT**+**A**.
   2. В диалоговом окне **Добавление существующего элемента** задайте для фильтра файлов значение **все файлы** или **файлы изображений**, перейдите к файлу изображения, который необходимо использовать, а затем нажмите кнопку **Добавить**.
   3. выберите файл изображения в **Обозреватель решений**, а затем в окне **свойства** задайте для параметра **действие сборки** значение **ресурс**.

**Создание и запуск приложения**

1. Чтобы выполнить сборку и запуск приложения, нажмите клавишу **F5** или выберите **начать отладку** в меню **Отладка** .

На следующем рисунке показано приложение с NavigationWindow кнопками:



1. Закройте приложение, чтобы вернуться к Visual Studio.

**Создание макета**

Макет предоставляет упорядоченный способ размещения элементов пользовательского интерфейса, а также управляет размером и положением этих элементов при изменении размера пользовательского интерфейса. Обычно макет создается с одним из следующих элементов управления макетом.

* Canvas — Определяет область, в которой можно явно располагать дочерние элементы с помощью координат, относящихся к области Canvas.
* DockPanel — Определяет область, в которой можно расположить дочерние элементы по горизонтали или по вертикали относительно друг друга.
* Grid — Определяет гибкую область сетки, состоящую из столбцов и строк.
* StackPanel — Упорядочивает дочерние элементы в одну строку, которая может быть ориентирована горизонтально или вертикально.
* VirtualizingStackPanel — Упорядочивает и виртуализировать содержимое в одной строке, ориентированной горизонтально или вертикально.
* WrapPanel — Размещает дочерние элементы в последовательном положении слева направо, разбивая содержимое на следующую строку на границе содержащего поля. Последующее упорядочение происходит последовательно сверху вниз или справа налево в зависимости от значения свойства Orientation.

Каждый из этих элементов управления макета поддерживает определенный тип макета для своих дочерних элементов. ExpenseIt размеры страниц можно изменять, и каждая страница содержит элементы, расположенные горизонтально и вертикально рядом с другими элементами. В этом примере Grid используется как элемент макета для приложения.

В этом разделе вы создадите таблицу с одним столбцом с тремя строками и 10-пиксельным полем, добавив определения Grid столбцов и строк в в *ExpenseItHome.xaml* .

1. В *ExpenseItHome.xaml* задайте Margin для свойства Grid элемента значение "10, 0, 10, 10", которое соответствует левому, верхнему, правому и нижнему полям:

XAML

<Grid Margin="10,0,10,10">

1. Добавьте следующий код XAML между Grid тегами, чтобы создать определения строк и столбцов:

XAML

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

Height Для двух строк устанавливается Auto значение, которое означает, что размер строк зависит от содержимого строк. По умолчанию Height используется Star размер, что означает, что высота строки является взвешенной пропорциями доступного пространства. Например, если две строки имеют Height "\*", каждая из них имеет высоту, равную половине доступного пространства.

Grid Теперь должен содержаться следующий код XAML:

XAML

<Grid Margin="10,0,10,10">

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

</Grid>

**Добавление элементов управления**

В этом разделе вы обновите пользовательский интерфейс домашней страницы, чтобы отобразить список людей, в которых вы выбрали одного пользователя для отображения отчета о расходах. Элементы управления — это объекты пользовательского интерфейса, позволяющие пользователям взаимодействовать с приложением.

Чтобы создать этот пользовательский интерфейс, добавьте следующие элементы в *ExpenseItHome.xaml* :

* ListBox (для списка пользователей).
* Label (для заголовка списка).
* Button (щелкните, чтобы просмотреть отчет о расходах для пользователя, выбранного в списке).

Каждый элемент управления помещается в строку объекта Grid путем установки Grid.Row присоединенного свойства..

1. В добавьте в *ExpenseItHome.xaml* тег следующий код XAML между [Grid](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.grid) тегами:

XAML

<!-- People list -->

<Border Grid.Column="0" Grid.Row="0" Height="35" Padding="5" Background="#4E87D4">

<Label VerticalAlignment="Center" Foreground="White">Names</Label>

</Border>

<ListBox Name="peopleListBox" Grid.Column="0" Grid.Row="1">

<ListBoxItem>Mike</ListBoxItem>

<ListBoxItem>Lisa</ListBoxItem>

<ListBoxItem>John</ListBoxItem>

<ListBoxItem>Mary</ListBoxItem>

</ListBox>

<!-- View report button -->

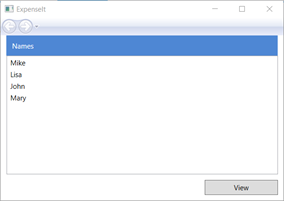
<Button Grid.Column="0" Grid.Row="2" Margin="0,10,0,0" Width="125" Height="25" HorizontalAlignment="Right">View</Button>

**Совет**

Можно также создать элементы управления, перетащив их из окна **панель элементов** в окно конструктора, а затем задав их свойства в окне **свойства** .

1. Создайте и запустите приложение.

На следующем рисунке показаны созданные элементы управления.



**Добавление изображения и заголовка**

В этом разделе вы обновите пользовательский интерфейс домашней страницы, используя изображение и заголовок страницы.

1. В *ExpenseItHome.xaml* добавьте еще один столбец ColumnDefinitions в с фиксированным Width 230 пикселов:

XAML

<Grid.ColumnDefinitions>

<ColumnDefinition Width="230" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

1. Добавьте еще одну строку в, чтобы [RowDefinitions](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.grid.rowdefinitions) всего четыре строки:

XAML

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="Auto"/>

<RowDefinition />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

1. Переместите элементы управления во второй столбец, задав Grid.Column свойству значение 1 в каждом из трех элементов управления (граница, ListBox и кнопка).
2. Переместите каждый элемент управления по строке, увеличив его Grid.Row значение на 1 для каждого из трех элементов управления (границы, ListBox и Button) и для элемента Border.

Теперь XAML для трех элементов управления выглядит следующим образом:

XAML

<Border Grid.Column="1" Grid.Row="1" Height="35" Padding="5" Background="#4E87D4">

<Label VerticalAlignment="Center" Foreground="White">Names</Label>

</Border>

<ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2">

<ListBoxItem>Mike</ListBoxItem>

<ListBoxItem>Lisa</ListBoxItem>

<ListBoxItem>John</ListBoxItem>

<ListBoxItem>Mary</ListBoxItem>

</ListBox>

<!-- View report button -->

<Button Grid.Column="1" Grid.Row="3" Margin="0,10,0,0" Width="125"

Height="25" HorizontalAlignment="Right">View</Button>

1. Задайте для свойства Background значение *watermark.png* файл изображения, ДОбавив следующий код XAML в любое место между <Grid> тегами и </Grid> :

XAML

<Grid.Background>

<ImageBrush ImageSource="watermark.png"/>

</Grid.Background>

1. Border Перед элементом добавьте Label с содержимым "Просмотр отчета о расходах". Эта метка является заголовком страницы.

XAML

<Label Grid.Column="1" VerticalAlignment="Center" FontFamily="Trebuchet MS"

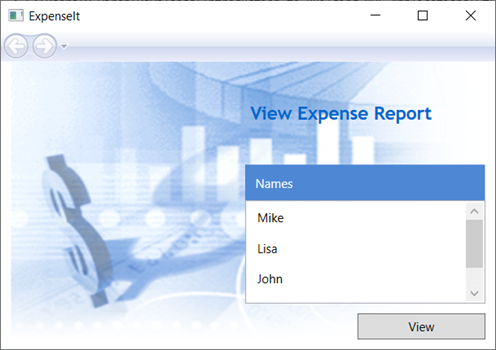
FontWeight="Bold" FontSize="18" Foreground="#0066cc">

View Expense Report

</Label>

1. Создайте и запустите приложение.

На следующем рисунке показаны результаты только что добавленных элементов.



**Добавление кода для обработчика событий**

1. В *ExpenseItHome.xaml* добавьте Click к Button элементу обработчик событий XAML

<!-- View report button -->

<Button Grid.Column="1" Grid.Row="3" Margin="0,10,0,0" Width="125"

Height="25" HorizontalAlignment="Right" Click="Button\_Click">View</Button>

1. Откройте  *ExpenseItHome.xaml.cs*.
2. Добавьте следующий код в ExpenseItHome класс, чтобы добавить обработчик событий нажатия кнопки. Обработчик событий открывает страницу **файл ExpenseReportPage** .

C#

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

{

// View Expense Report

ExpenseReportPage expenseReportPage = new ExpenseReportPage();

this.NavigationService.Navigate(expenseReportPage);

}

**Создание пользовательского интерфейса для файла ExpenseReportPage**

*Файл ExpenseReportPage. XAML* отображает отчет о расходах для пользователя, выбранного на **ExpenseItHome** странице. В этом разделе вы создадите пользовательский интерфейс для **файла ExpenseReportPage**. Вы также добавите цвета фона и заливки в различные элементы пользовательского интерфейса.

1. Откройте *файл ExpenseReportPage. XAML*.
2. Добавьте следующий код XAML между Grid тегами:

XAML

<Grid.Background>

<ImageBrush ImageSource="watermark.png" />

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="230" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<Label Grid.Column="1" VerticalAlignment="Center" FontFamily="Trebuchet MS"

FontWeight="Bold" FontSize="18" Foreground="#0066cc">

Expense Report For:

</Label>

<Grid Margin="10" Grid.Column="1" Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<!-- Name -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Orientation="Horizontal">

<Label Margin="0,0,0,5" FontWeight="Bold">Name:</Label>

<Label Margin="0,0,0,5" FontWeight="Bold"></Label>

</StackPanel>

<!-- Department -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Orientation="Horizontal">

<Label Margin="0,0,0,5" FontWeight="Bold">Department:</Label>

<Label Margin="0,0,0,5" FontWeight="Bold"></Label>

</StackPanel>

<Grid Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="2" VerticalAlignment="Top"

HorizontalAlignment="Left">

<!-- Expense type and Amount table -->

<DataGrid AutoGenerateColumns="False" RowHeaderWidth="0" >

<DataGrid.ColumnHeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}">

<Setter Property="Height" Value="35" />

<Setter Property="Padding" Value="5" />

<Setter Property="Background" Value="#4E87D4" />

<Setter Property="Foreground" Value="White" />

</Style>

</DataGrid.ColumnHeaderStyle>

<DataGrid.Columns>

<DataGridTextColumn Header="ExpenseType" />

<DataGridTextColumn Header="Amount" />

</DataGrid.Columns>

</DataGrid>

</Grid>

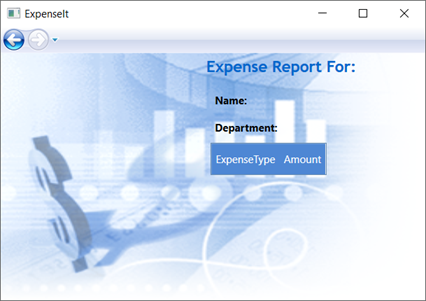
</Grid>

Этот пользовательский интерфейс аналогичен *ExpenseItHome.xaml* , за исключением того, что данные отчета отображаются в DataGrid .

1. Создайте и запустите приложение.
2. Нажмите кнопку **Просмотр** .

Появится страница отчета по расходам. Также обратите внимание, что кнопка обратной навигации включена.

На следующем рисунке показаны элементы пользовательского интерфейса, добавленные в *файл ExpenseReportPage. XAML*.



**Элементы управления стиля**

Внешний вид различных элементов часто одинаков для всех элементов одного типа в пользовательском интерфейсе. Пользовательский интерфейс использует стили для того, чтобы внешний вид можно было использовать в нескольких элементах. Многократное использование стилей помогает упростить создание XAML и управление им. В этом разделе атрибуты, установленные ранее для каждого элемента, заменяются стилями.

1. Откройте *Application. XAML* или *app. XAML*.
2. Добавьте следующий код XAML между Application.Resources тегами:

XAML

<!-- Header text style -->

<Style x:Key="headerTextStyle">

<Setter Property="Label.VerticalAlignment" Value="Center"></Setter>

<Setter Property="Label.FontFamily" Value="Trebuchet MS"></Setter>

<Setter Property="Label.FontWeight" Value="Bold"></Setter>

<Setter Property="Label.FontSize" Value="18"></Setter>

<Setter Property="Label.Foreground" Value="#0066cc"></Setter>

</Style>

<!-- Label style -->

<Style x:Key="labelStyle" TargetType="{x:Type Label}">

<Setter Property="VerticalAlignment" Value="Top" />

<Setter Property="HorizontalAlignment" Value="Left" />

<Setter Property="FontWeight" Value="Bold" />

<Setter Property="Margin" Value="0,0,0,5" />

</Style>

<!-- DataGrid header style -->

<Style x:Key="columnHeaderStyle" TargetType="{x:Type DataGridColumnHeader}">

<Setter Property="Height" Value="35" />

<Setter Property="Padding" Value="5" />

<Setter Property="Background" Value="#4E87D4" />

<Setter Property="Foreground" Value="White" />

</Style>

<!-- List header style -->

<Style x:Key="listHeaderStyle" TargetType="{x:Type Border}">

<Setter Property="Height" Value="35" />

<Setter Property="Padding" Value="5" />

<Setter Property="Background" Value="#4E87D4" />

</Style>

<!-- List header text style -->

<Style x:Key="listHeaderTextStyle" TargetType="{x:Type Label}">

<Setter Property="Foreground" Value="White" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="HorizontalAlignment" Value="Left" />

</Style>

<!-- Button style -->

<Style x:Key="buttonStyle" TargetType="{x:Type Button}">

<Setter Property="Width" Value="125" />

<Setter Property="Height" Value="25" />

<Setter Property="Margin" Value="0,10,0,0" />

<Setter Property="HorizontalAlignment" Value="Right" />

</Style>

Этот код XAML добавляет следующие стили:

* + headerTextStyleдля форматирования заголовка страницы Label;
  + labelStyleдля форматирования элементов управления Label ;
  + columnHeaderStyleдля форматирования DataGridColumnHeader;
  + listHeaderStyleдля форматирования элементов управления Border заголовков списка;
  + listHeaderTextStyle— Форматирование заголовка Label списка.
  + buttonStyle— Для форматирования Button ExpenseItHome.xaml .

Обратите внимание, что стили являются ресурсами и дочерними элементами Application.Resources элемента Property. Здесь стили применяются ко всем элементам в приложении. В *ExpenseItHome.xaml* замените все между Grid элементами следующим кодом XAML:

XAML

<Grid.Background>

<ImageBrush ImageSource="watermark.png" />

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="230" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="Auto"/>

<RowDefinition />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<!-- People list -->

<Label Grid.Column="1" Style="{StaticResource headerTextStyle}" >

View Expense Report

</Label>

<Border Grid.Column="1" Grid.Row="1" Style="{StaticResource listHeaderStyle}">

<Label Style="{StaticResource listHeaderTextStyle}">Names</Label>

</Border>

<ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2">

<ListBoxItem>Mike</ListBoxItem>

<ListBoxItem>Lisa</ListBoxItem>

<ListBoxItem>John</ListBoxItem>

<ListBoxItem>Mary</ListBoxItem>

</ListBox>

<!-- View report button -->

<Button Grid.Column="1" Grid.Row="3" Click="Button\_Click" Style="{StaticResource buttonStyle}">View</Button>

Свойства, определяющие внешний вид элементов управления, такие как VerticalAlignment и FontFamily , при применении стилей удаляются и заменяются. Например, headerTextStyle применяется к «Просмотр отчета о расходах» Label .

1. Откройте *файл ExpenseReportPage. XAML*.
2. Замените все Grid элементы на следующий код XAML:

XAML

<Grid.Background>

<ImageBrush ImageSource="watermark.png" />

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="230" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<Label Grid.Column="1" Style="{StaticResource headerTextStyle}">

Expense Report For:

</Label>

<Grid Margin="10" Grid.Column="1" Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<!-- Name -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Orientation="Horizontal">

<Label Style="{StaticResource labelStyle}">Name:</Label>

<Label Style="{StaticResource labelStyle}"></Label>

</StackPanel>

<!-- Department -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1"

Orientation="Horizontal">

<Label Style="{StaticResource labelStyle}">Department:</Label>

<Label Style="{StaticResource labelStyle}"></Label>

</StackPanel>

<Grid Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="2" VerticalAlignment="Top"

HorizontalAlignment="Left">

<!-- Expense type and Amount table -->

<DataGrid ColumnHeaderStyle="{StaticResource columnHeaderStyle}"

AutoGenerateColumns="False" RowHeaderWidth="0" >

<DataGrid.Columns>

<DataGridTextColumn Header="ExpenseType" />

<DataGridTextColumn Header="Amount" />

</DataGrid.Columns>

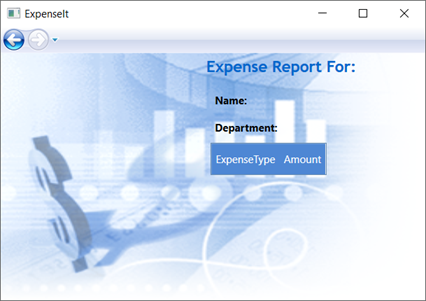
</DataGrid>

</Grid>

</Grid>

Этот XAML добавляет стили в Label элементы и Border .

1. Создайте и запустите приложение. Внешний вид окна такой же, как и ранее.



1. Закройте приложение, чтобы вернуться к Visual Studio.

**Привязка данных к элементу управления**

В этом разделе вы создадите XML-данные, привязанные к различным элементам управления.

1. В *ExpenseItHome.xaml* после открывающего Grid элемента добавьте следующий код XAML, чтобы создать объект XmlDataProvider , содержащий данные для каждого пользователя:

XAMLКопировать

<Grid.Resources>

<!-- Expense Report Data -->

<XmlDataProvider x:Key="ExpenseDataSource" XPath="Expenses">

<x:XData>

<Expenses xmlns="">

<Person Name="Mike" Department="Legal">

<Expense ExpenseType="Lunch" ExpenseAmount="50" />

<Expense ExpenseType="Transportation" ExpenseAmount="50" />

</Person>

<Person Name="Lisa" Department="Marketing">

<Expense ExpenseType="Document printing"

ExpenseAmount="50"/>

<Expense ExpenseType="Gift" ExpenseAmount="125" />

</Person>

<Person Name="John" Department="Engineering">

<Expense ExpenseType="Magazine subscription"

ExpenseAmount="50"/>

<Expense ExpenseType="New machine" ExpenseAmount="600" />

<Expense ExpenseType="Software" ExpenseAmount="500" />

</Person>

<Person Name="Mary" Department="Finance">

<Expense ExpenseType="Dinner" ExpenseAmount="100" />

</Person>

</Expenses>

</x:XData>

</XmlDataProvider>

</Grid.Resources>

Данные создаются в виде Grid ресурса. Обычно эти данные загружаются в виде файла, но для простоты данные добавляются встроенным образом.

1. <Grid.Resources>В элементе добавьте следующий <xref:System.Windows.DataTemplate> элемент, определяющий способ отображения данных в ListBox , после <XmlDataProvider> элемента:

XAMLКопировать

<Grid.Resources>

<!-- Name item template -->

<DataTemplate x:Key="nameItemTemplate">

<Label Content="{Binding XPath=@Name}"/>

</DataTemplate>

</Grid.Resources>

1. Замените существующий ListBox код XAML следующим:

XAML

<ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2"

ItemsSource="{Binding Source={StaticResource ExpenseDataSource}, XPath=Person}"

ItemTemplate="{StaticResource nameItemTemplate}">

</ListBox>

Этот XAML привязывает ItemsSource свойство объекта ListBox к источнику данных и применяет шаблон данных в качестве ItemTemplate .

**Подключение данных в элементы управления**

Далее предстоит добавить код для получения имени, выбранного на **ExpenseItHome** странице, и передачи его конструктору **файл ExpenseReportPage**. **Файл ExpenseReportPage** устанавливает в качестве контекста данных переданный элемент, который является элементом управления, определенным в привязке *файл ExpenseReportPage. XAML*.

1. Откройте файл *ExpenseReportPage.xaml.vb* или *ExpenseReportPage.xaml.cs*.
2. Добавьте конструктор, принимающий объект, чтобы можно было передавать данные отчета о затратах выбранного человека.

C#

public partial class ExpenseReportPage : Page

{

public ExpenseReportPage()

{

InitializeComponent();

}

// Custom constructor to pass expense report data

public ExpenseReportPage(object data):this()

{

// Bind to expense report data.

this.DataContext = data;

}

}

1. Откройте  *ExpenseItHome.xaml.cs*.
2. Click Измените обработчик событий, чтобы вызвать новый конструктор, передающий данные отчета о затратах выбранного человека.

C#

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

{

// View Expense Report

ExpenseReportPage expenseReportPage = new ExpenseReportPage(this.peopleListBox.SelectedItem);

this.NavigationService.Navigate(expenseReportPage);

}

**Применение стилей к данным с помощью шаблонов данных**

В этом разделе вы обновите пользовательский интерфейс для каждого элемента в списках с привязкой к данным с помощью шаблонов данных.

1. Откройте *файл ExpenseReportPage. XAML*.
2. Привяжите содержимое элементов "Name" и "Department" Label к соответствующему свойству источника данных. XAML

<!-- Name -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Orientation="Horizontal">

<Label Style="{StaticResource labelStyle}">Name:</Label>

<Label Style="{StaticResource labelStyle}" Content="{Binding XPath=@Name}"></Label>

</StackPanel>

<!-- Department -->

<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Orientation="Horizontal">

<Label Style="{StaticResource labelStyle}">Department:</Label>

<Label Style="{StaticResource labelStyle}" Content="{Binding XPath=@Department}"></Label>

</StackPanel>

1. После открывающего Grid элемента добавьте следующие шаблоны данных, определяющие способ отображения данных отчета о расходах:

XAML

<!--Templates to display expense report data-->

<Grid.Resources>

<!-- Reason item template -->

<DataTemplate x:Key="typeItemTemplate">

<Label Content="{Binding XPath=@ExpenseType}"/>

</DataTemplate>

<!-- Amount item template -->

<DataTemplate x:Key="amountItemTemplate">

<Label Content="{Binding XPath=@ExpenseAmount}"/>

</DataTemplate>

</Grid.Resources>

1. DataGridTextColumn Замените элементы DataGridTemplateColumn элементом в DataGrid элементе и примените к ним шаблоны. Кроме того, укажите ItemsSource атрибут со значением в DataGrid элементе.

XAML

<!-- Expense type and Amount table -->

<DataGrid ItemsSource="{Binding XPath=Expense}" ColumnHeaderStyle="{StaticResource columnHeaderStyle}" AutoGenerateColumns="False" RowHeaderWidth="0" >

<DataGrid.Columns>

<DataGridTemplateColumn Header="ExpenseType" CellTemplate="{StaticResource typeItemTemplate}" />

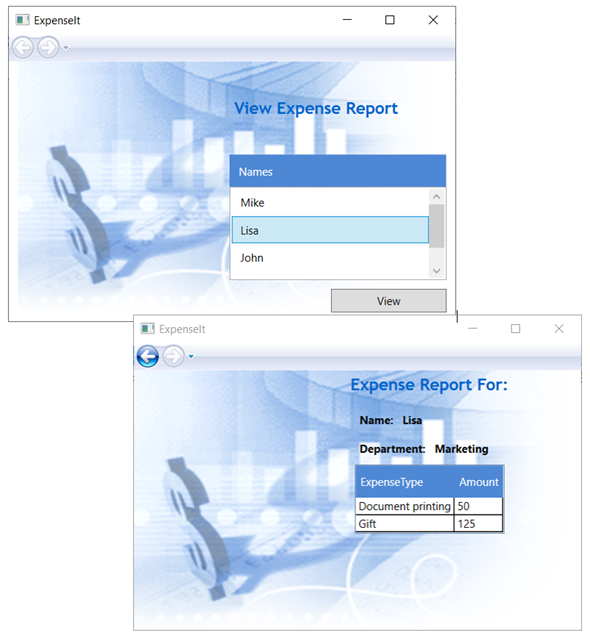
<DataGridTemplateColumn Header="Amount" CellTemplate="{StaticResource amountItemTemplate}" />

</DataGrid.Columns>

</DataGrid>

1. Создайте и запустите приложение.
2. Выберите пользователя и нажмите кнопку **Просмотр** .

На следующем рисунке показаны обе страницы ExpenseIt приложения с примененными элементами управления, макет, стили, привязка данных и шаблоны данных:



**Примечание**

В этом примере демонстрируется конкретная функция WPF, которая не соответствует всем рекомендациям, таким как безопасность, локализация и специальные возможности.

**Дальнейшие действия**

в этом пошаговом руководстве вы узнали о ряде методов создания пользовательского интерфейса с помощью Windows Presentation Foundation (WPF). Теперь у вас должно быть базовое представление о стандартных блоках приложений .NET с привязкой к данным.