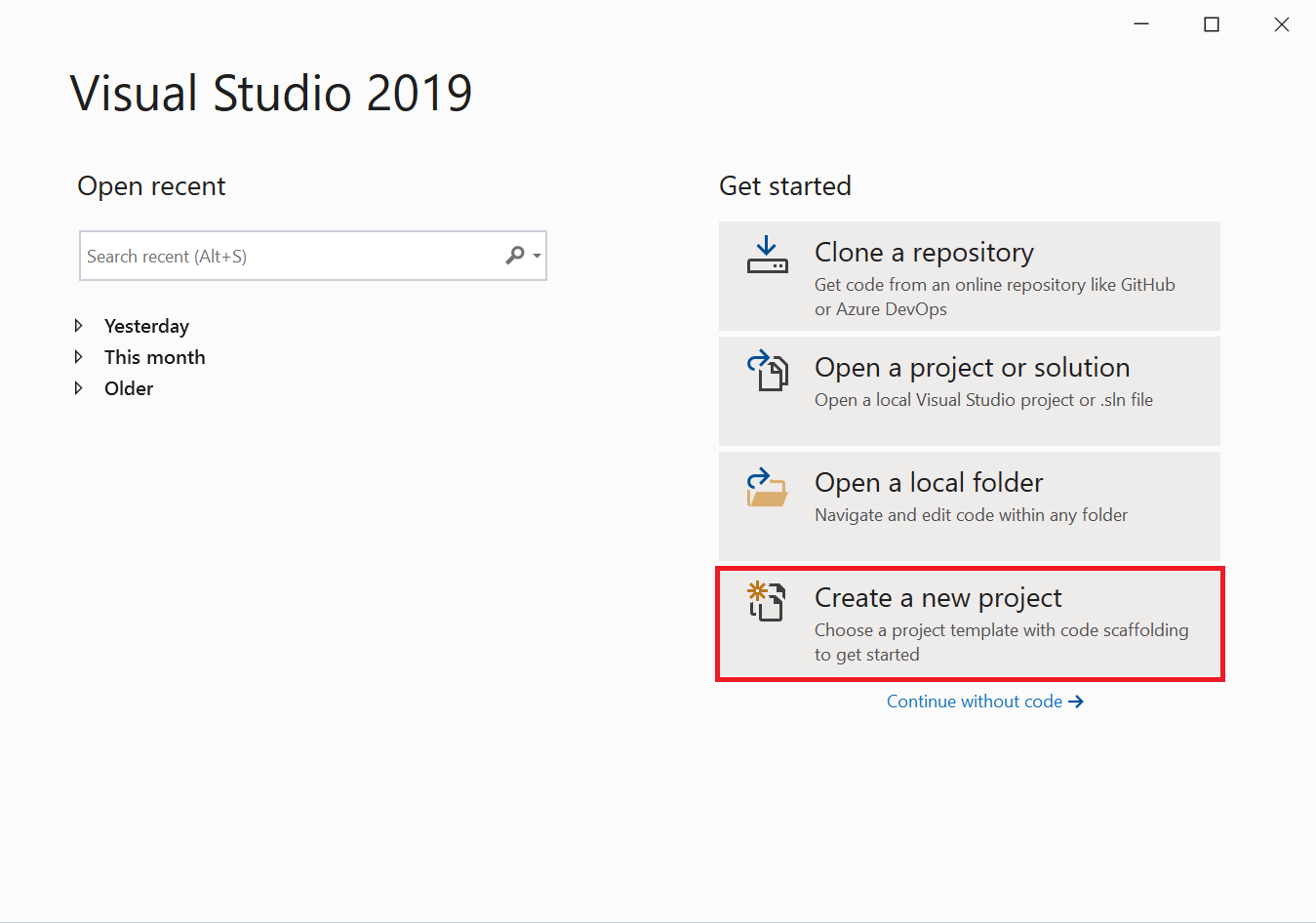
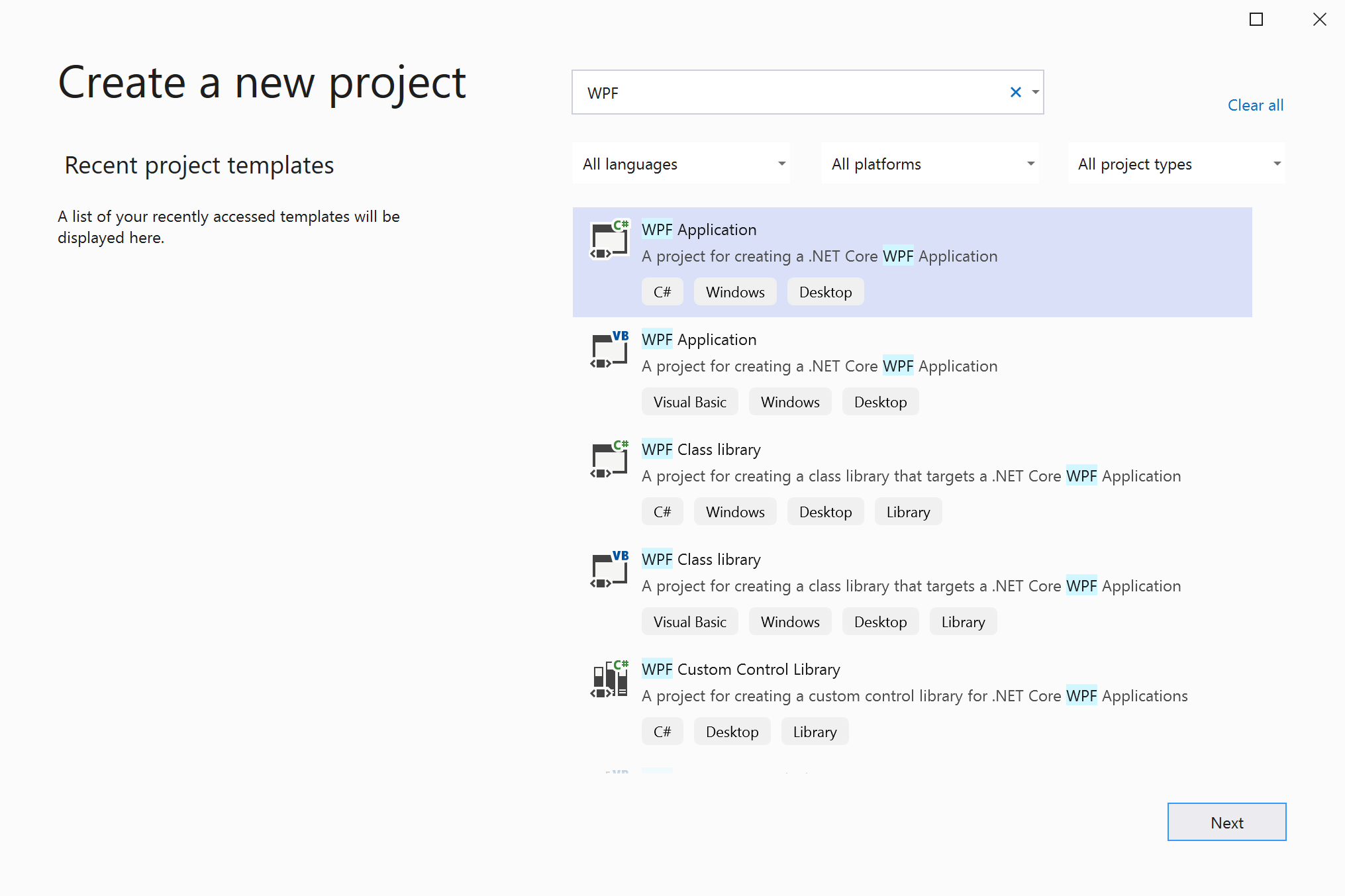
# Создание, сборка и отладка простого приложения на C#

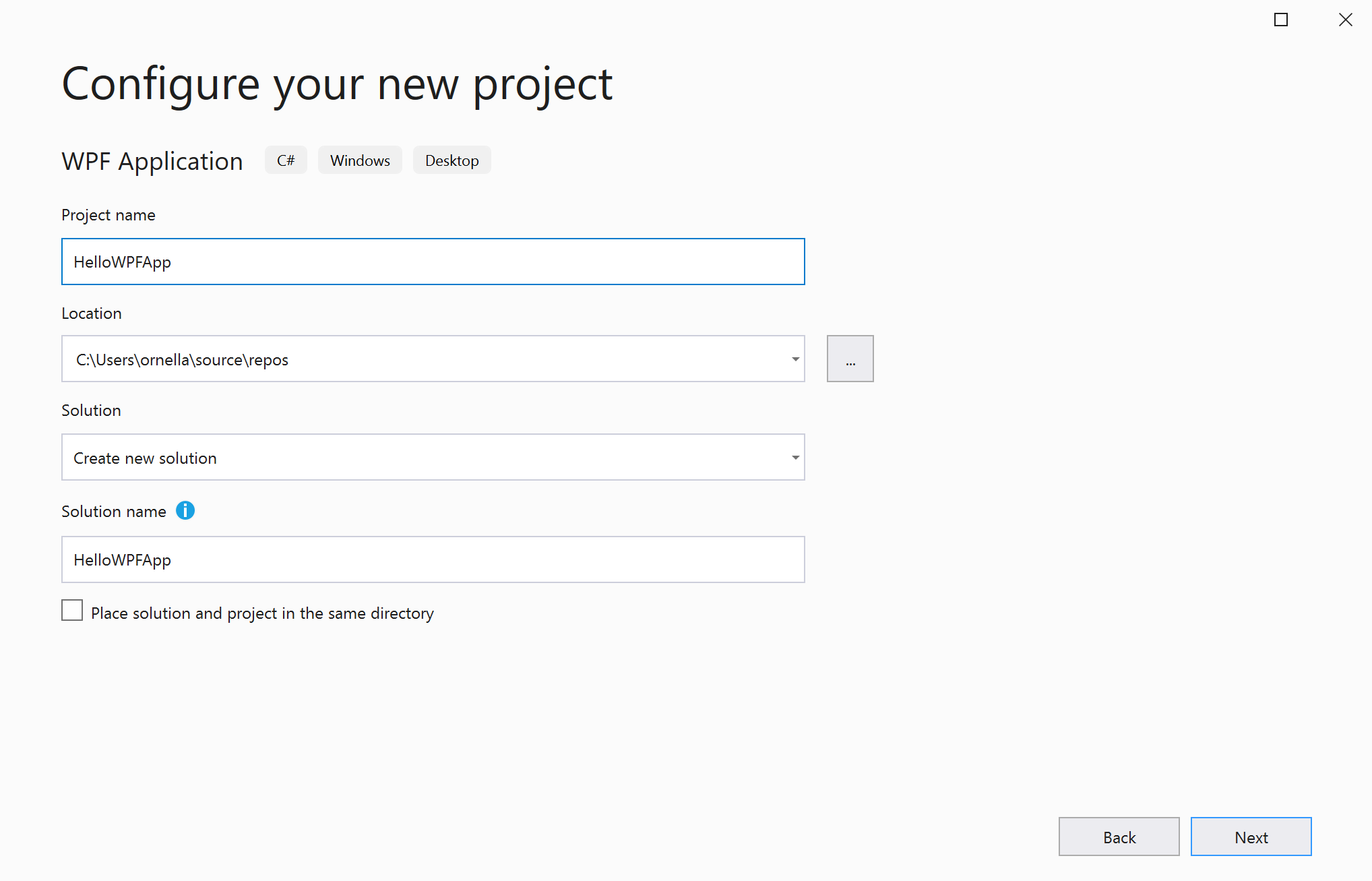
1. Запустите Visual Studio.
2. На начальном экране выберите **Создать проект**.



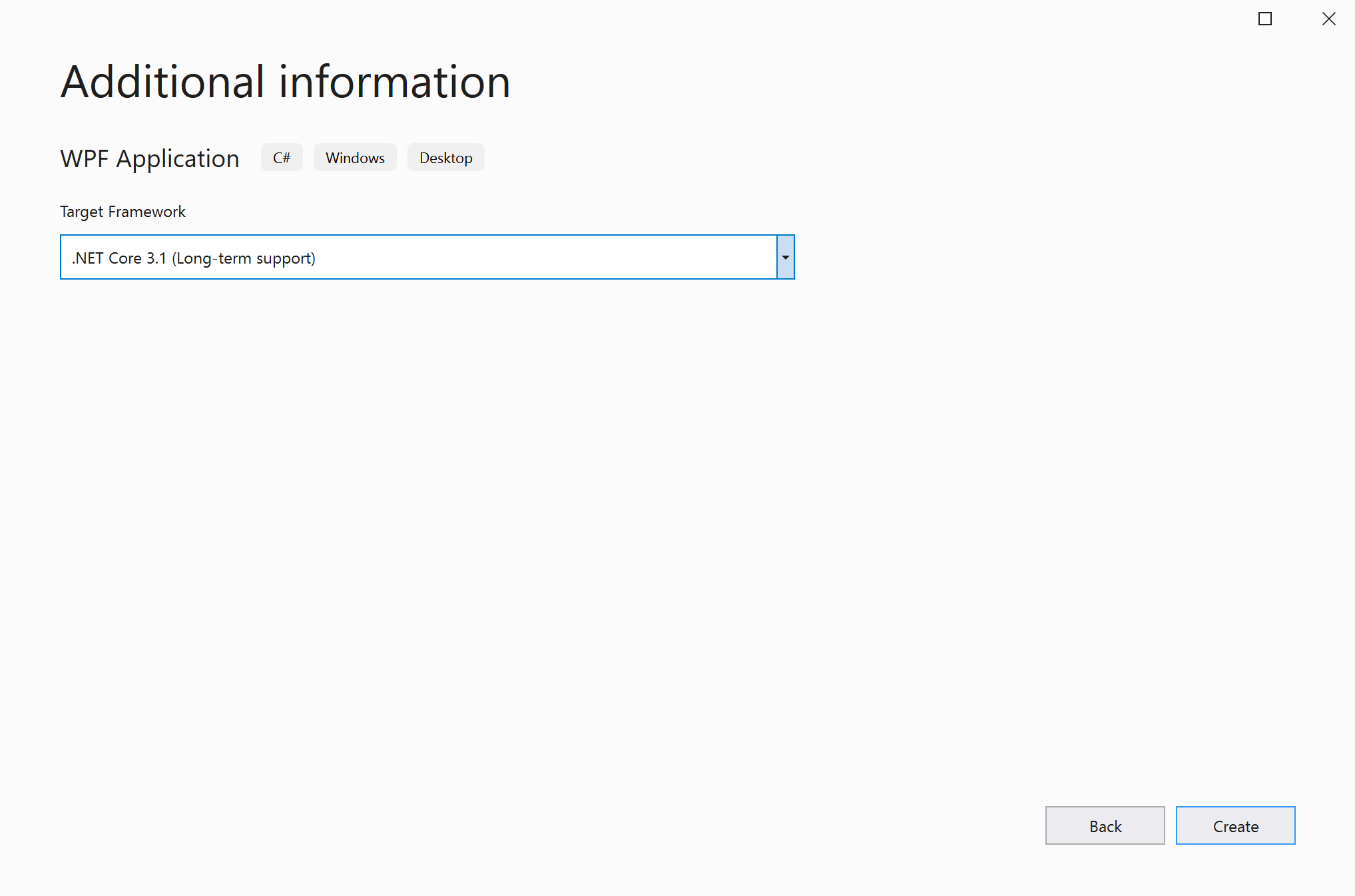
1. На экране **Создание проекта** выполните поиск по запросу "WPF", выберите в результатах **Приложение WPF** и нажмите кнопку **Далее**.



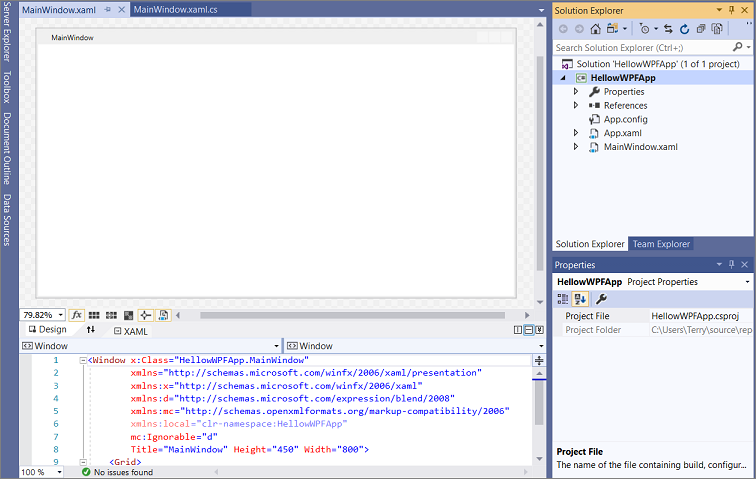
1. На следующем экране присвойте проекту имя **HelloWPFApp** и щелкните **Далее**.



1. В окне **Дополнительные сведения** для целевой платформы должна быть указана версия **.NET Core 3.1**. Если это не так, выберите **.NET Core 3.1**. Затем нажмите **Создать**.

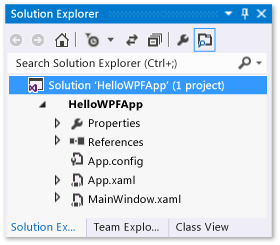


Visual Studio создает решение и проект HelloWPFApp, а в **обозревателе решений** выводятся различные файлы. **Конструктор WPF** отображает представление кода и представление XAML MainWindow.xaml в представлении с разделением. Сдвигая разделитель, можно делать любое из представлений больше или меньше. Можно выбрать для просмотра только визуальное представление или только представление XAML.



**Примечание**

После создания проекта его можно настраивать. Для этого выберите **Окно свойств** в меню **Представление** или нажмите клавишу **F4**. Затем можно отображать и изменять параметры элементов проекта, элементов управления и других элементов в приложении.



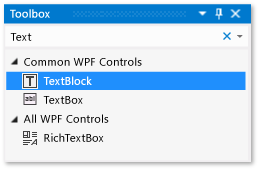
## Конструирование пользовательского интерфейса (ИП)

Если конструктор не открыт, выберите файл MainWindow.xaml и нажмите клавиши **SHIFT**+**F7**, чтобы открыть его.

В приложение будут добавлены элементы управления трех типов: элемент управления TextBlock, два элемента управления RadioButton и элемент управления Button.

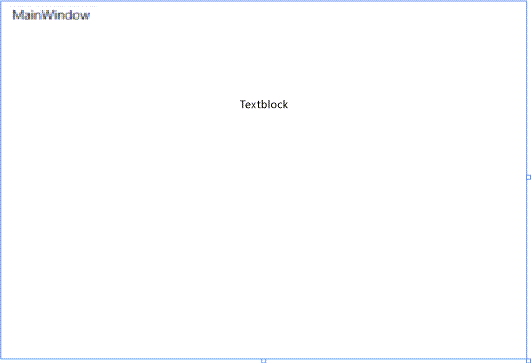
### Добавление элемента управления TextBlock

1. Нажмите клавиши **CTRL**+**Q** для активации поля поиска и введите **Панель элементов**. Выберите в списке результатов **Представление > Панель элементов**.
2. В окне **Панель элементов** разверните узел **Типовые элементы управления WPF**, чтобы увидеть элемент управления TextBlock.



1. Добавьте элемент управления TextBlock в область конструктора, выбрав элемент управления **TextBlock** и перетащив его в окно. Отцентрируйте этот элемент в верхней части окна. В Visual Studio 2019 и более поздних версиях для выравнивания элемента управления по центру можно использовать красные направляющие.

Окно должно выглядеть так, как показано на следующем рисунке:



Разметка XAML должна выглядеть приблизительно так, как в следующем примере:

XAML

<Grid>

<TextBlock HorizontalAlignment="Left" Margin="387,60,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top"/>

</Grid>

### Настройка текста в текстовом блоке

1. В представлении XAML найдите разметку для **TextBlock** и измените атрибут **Text** с TextBox на Select a message option and then choose the Display button..

Разметка XAML должна выглядеть приблизительно так, как в следующем примере:

XAMLКопировать

<Grid>

<TextBlock HorizontalAlignment="Left" Margin="387,60,0,0" TextWrapping="Wrap" Text="Select a message option and then choose the Display button." VerticalAlignment="Top"/>

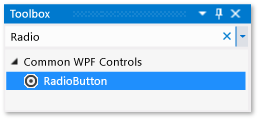
</Grid>

1. При необходимости снова разместите элемент TextBlock по центру и сохраните изменения сочетанием клавиш **CTRL+S** или с помощью пункта меню **Файл**.

Следующий шаг — добавить в форму два элемента управления RadioButton.

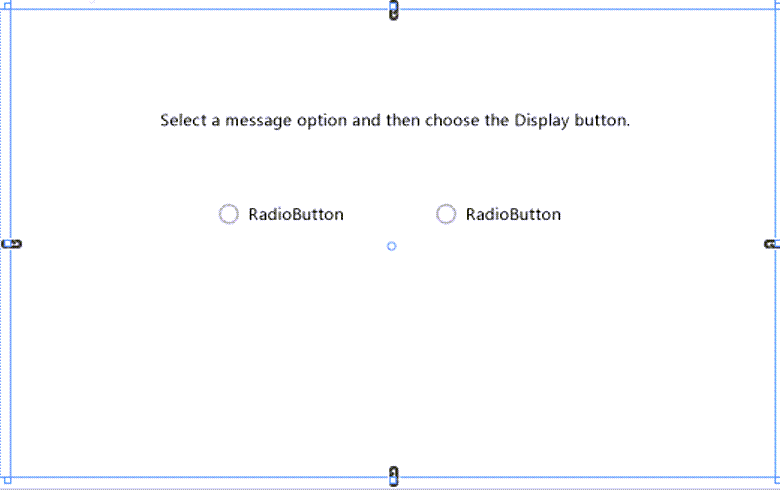
### Добавление переключателей

1. На **панели элементов** найдите элемент управления **RadioButton**.

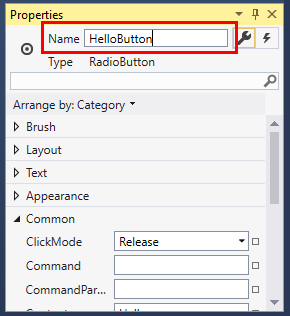


1. Добавьте два элемента управления RadioButton в область конструктора, выбрав элемент **RadioButton** и перетащив его в область конструктора. Переместите кнопки (выбрав их и используя клавиши со стрелками), чтобы кнопки отображались рядом под элементом управления TextBlock. Используйте красные направляющие для выравнивания элемента управления.

Окно должно выглядеть следующим образом:



1. В окне **Свойства** для левого элемента управления RadioButton измените свойство **Имя** (свойство в верхней части окна **Свойства** ), задав ему значение HelloButton.



1. В окне **Свойства** правого элемента управления RadioButton измените свойство **Имя**, задав ему значение GoodbyeButton, и затем сохраните изменения.

Теперь вы добавите отображаемый текст для каждого элемента управления RadioButton. Следующая процедура обновляет свойство **Content** элемента управления RadioButton.

### Добавление отображаемого текста для каждого переключателя

1. Измените атрибут **Content** для кнопок HelloButton и GoodbyeButton на "Hello" и "Goodbye" в XAML. Разметка XAML теперь должна выглядеть следующим образом:

XAML

<Grid>

<TextBlock HorizontalAlignment="Left" Margin="252,47,0,0" TextWrapping="Wrap" Text="Select a message option and then choose the Display button." VerticalAlignment="Top"/>

<RadioButton x:Name="HelloButton" Content="Hello" HorizontalAlignment="Left" Margin="297,161,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left" Margin="488,161,0,0" VerticalAlignment="Top"/>

</Grid>

### Задание переключателя, выбранного по умолчанию

На этом этапе мы настроим элемент HelloButton как выбранный по умолчанию, чтобы всегда был выбран один из переключателей.

1. В представлении XAML найдите разметку для элемента HelloButton.
2. Добавьте атрибут **IsChecked** и задайте для него значение **True**. В частности, добавьте IsChecked="True".

Разметка XAML теперь должна выглядеть следующим образом:

XAML

<Grid>

<TextBlock HorizontalAlignment="Left" Margin="252,47,0,0" TextWrapping="Wrap" Text="Select a message option and then choose the Display button." VerticalAlignment="Top"/>

<RadioButton x:Name="HelloButton" Content="Hello" IsChecked="True" HorizontalAlignment="Left" Margin="297,161,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left" Margin="488,161,0,0" VerticalAlignment="Top"/>

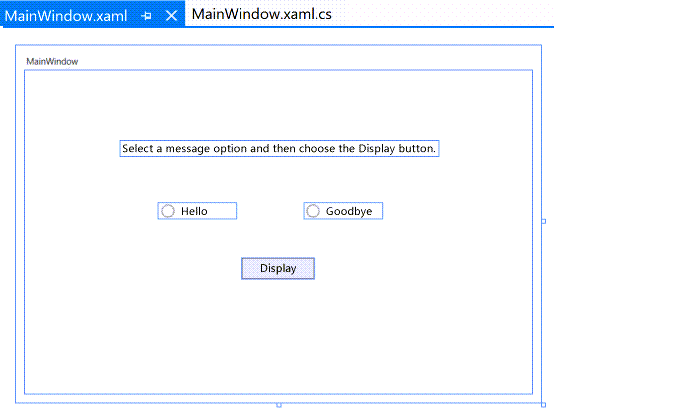
</Grid>

Последний элемент пользовательского интерфейса, который вам предстоит добавить, — это Button.

### Добавление элемента управления Button

1. На **панели элементов** найдите элемент управления **Button** и добавьте его в область конструктора под элементами управления RadioButton, перетащив его на форму в представлении конструирования. Если вы используете Visual Studio 2019 или более позднюю версию, выровнять элемент управления по центру можно с помощью красной линии.
2. В представлении XAML измените значение свойства **Содержимое** элемента управления Button с Content="Button" на Content="Display" и сохраните изменения.

Окно должно выглядеть так, как показано на следующем рисунке.



Разметка XAML теперь должна выглядеть следующим образом:

XAML

<Grid>

<TextBlock HorizontalAlignment="Left" Margin="252,47,0,0" TextWrapping="Wrap" Text="Select a message option and then choose the Display button." VerticalAlignment="Top"/>

<RadioButton x:Name="HelloButton" Content="Hello" IsChecked="True" HorizontalAlignment="Left" Margin="297,161,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left" Margin="488,161,0,0" VerticalAlignment="Top"/>

<Button Content="Display" HorizontalAlignment="Left" Margin="377,270,0,0" VerticalAlignment="Top" Width="75"/>

</Grid>

### Добавление кода к кнопке Display

После запуска приложения окно сообщения появится только тогда, когда пользователь выберет переключатель, а затем нажмет кнопку **Display**. Одно окно сообщения появится для Hello, и другое — для Goodbye. Чтобы задать такое поведение, добавьте код в событие Button\_Click в файле Greetings.xaml.cs.

1. На поверхности разработки дважды щелкните кнопку **Display** .

Откроется файл Greetings.xaml.cs, а курсор будет установлен на событии Button\_Click.

C#Копировать

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

{

}

1. Введите следующий код:

C#Копировать

if (HelloButton.IsChecked == true)

{

MessageBox.Show("Hello.");

}

else if (GoodbyeButton.IsChecked == true)

{

MessageBox.Show("Goodbye.");

}

1. Сохраните приложение.

## Отладка и тестирование приложения

После этого вам предстоит отладить приложение для выявления ошибок и тестирования того, что оба окна сообщений отображаются правильно. Приведенные ниже инструкции описывают, как выполнить сборку и запустить отладчик

### Изменение имени MainWindow.xaml

Давайте присвоим файлу MainWindow более конкретное имя. В **обозревателе решений** щелкните правой кнопкой мыши файл MainWindow.xaml и выберите команду **Переименовать**. Измените имя файла на Greetings.xaml.

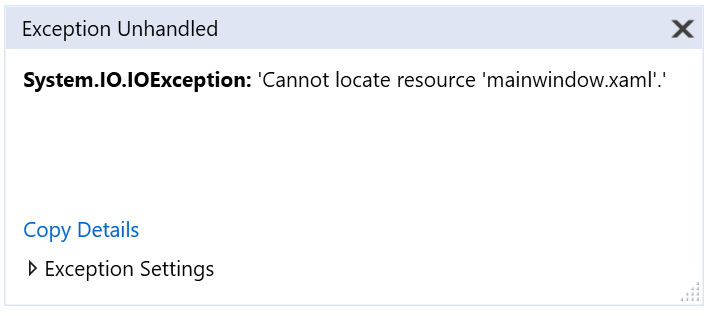
### Поиск и исправление ошибок

В этом шаге вам предстоит найти ошибку, которую мы намеренно допустили ранее, изменив имя файла MainWindow.xaml.

#### Начало отладки и поиск ошибки

1. Запустите отладчик, нажав клавишу **F5** или выбрав пункты меню **Отладка** и **Начать отладку**.

Появится окно **Режим приостановки выполнения**, а в окне **Вывод** будет указано, что произошло исключение IOException: "Не удается найти ресурс mainwindow.xaml".



1. Остановите отладчик, выбрав **Отладка** > **Остановить отладку**.

Мы переименовали файл MainWindow.xaml в Greetings.xaml, но код приложения по-прежнему ссылается на MainWindow.xaml в качестве универсального кода ресурса (URI), поэтому проект не может быть запущен.

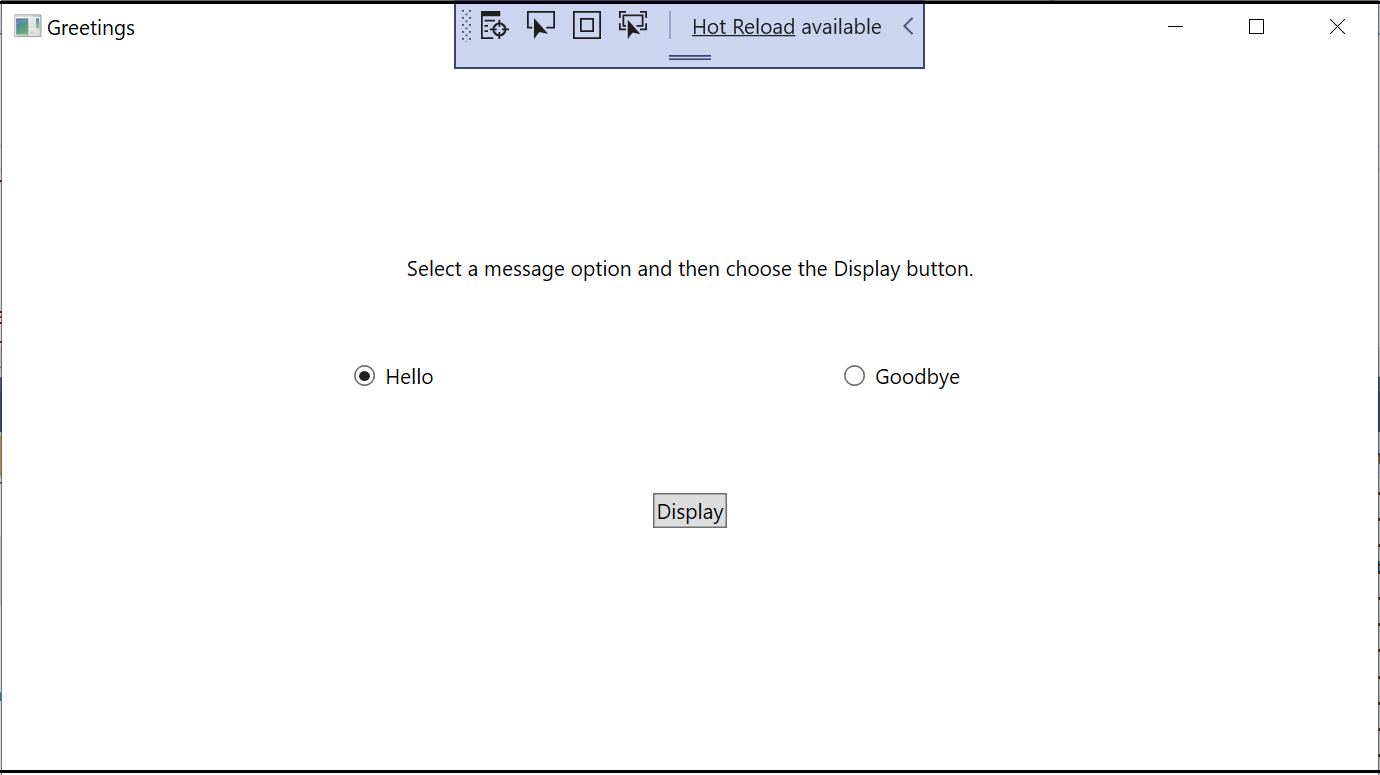
#### Задание Greetings.xaml в качестве начального универсального кода ресурса (URI)

1. В **обозревателе решений** откройте файл App.xaml.
2. Измените StartupUri="MainWindow.xaml" на StartupUri="Greetings.xaml" и сохраните изменения.

Как дополнительный шаг это позволит избежать путаницы при изменении заголовка окна приложения в соответствии с новым именем.

1. В **Обозревателе решений** откройте файл Greetings.xaml, который вы только что переименовали.
2. Измените значение свойства **Window.Title** с Title="MainWindow" на Title="Greetings" и сохраните изменения.

Запустите отладчик снова (клавишей **F5**). Должно появиться окно **приветствия** приложения.



Теперь закройте окно приложения, чтобы остановить отладку.

### Отладка с точками останова

Добавив точки останова, можно тестировать код во время отладки. Точки останова можно добавлять, выбирая в меню **Отладка** > **Точка останова**, щелкая в левом поле редактора рядом со строкой кода, в которой требуется приостановить выполнение, или нажимая клавишу **F9**.

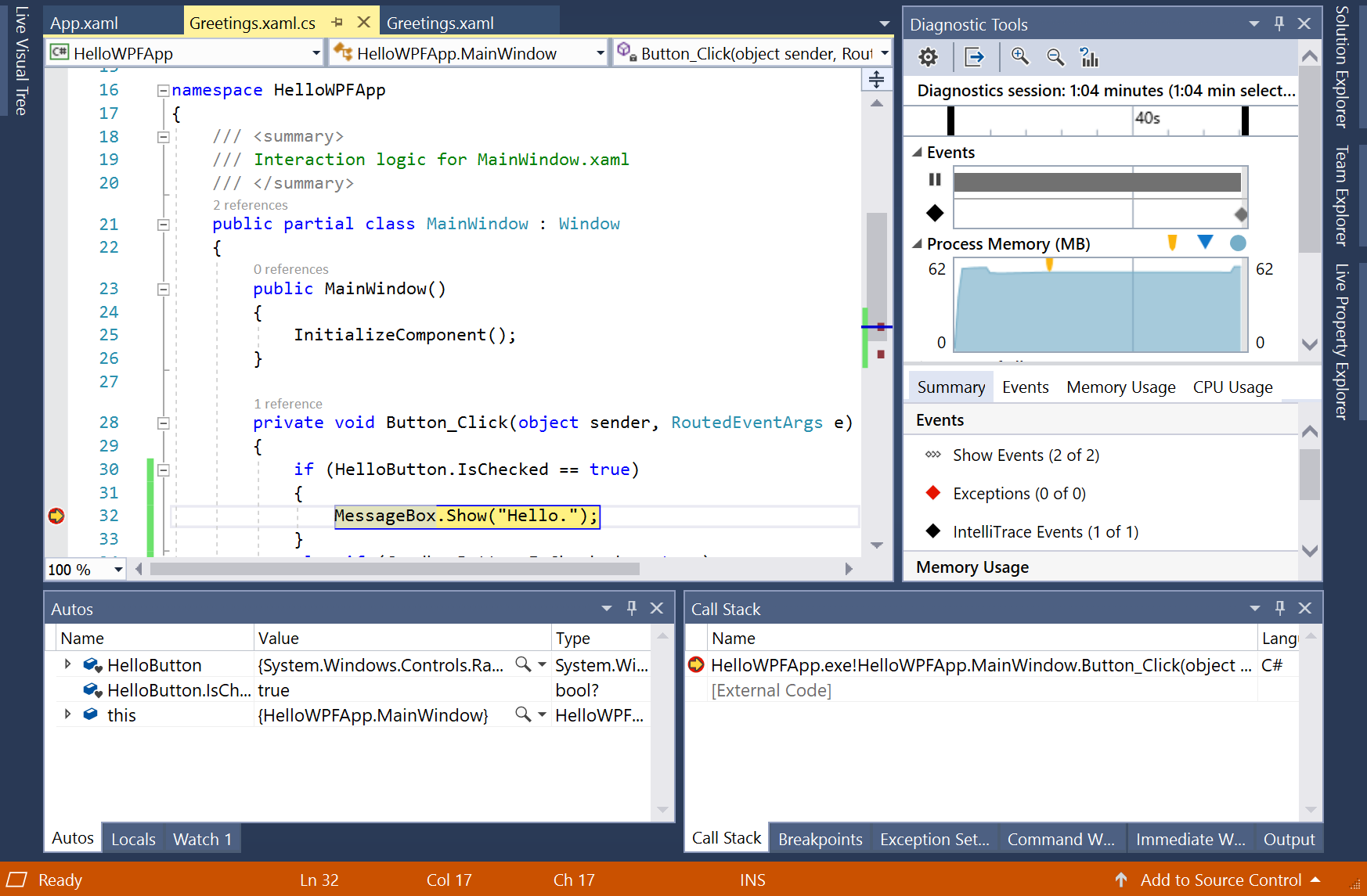
#### Добавление точек останова

1. Откройте файл Greetings.xaml.cs и выделите строку MessageBox.Show("Hello.").
2. Добавьте точку останова, выбрав меню **Отладка**, затем — **Точка останова**.

Рядом со строкой кода в крайнем левом поле окна редактора появится красный кружок.

1. Выделите следующую строку: MessageBox.Show("Goodbye.").
2. Нажмите клавишу **F9**, чтобы добавить точку останова, а затем нажмите клавишу **F5**, чтобы начать отладку.
3. В окне **Greetings** выберите переключатель **Hello** и нажмите кнопку **Display** .

Строка MessageBox.Show("Hello.") выделяется желтым цветом. В нижней части интегрированной среды разработки слева закреплены окна "Видимые", "Локальные" и "Контрольные значения", а справа — окна "Стек вызовов", "Точки останова", "Параметры исключений", "Команда", "Интерпретация" и "Вывод".



1. В строке меню выберите **Отладка** > **Шаг с выходом**.

Приложение возобновит выполнение, и появится окно сообщения со словом "Hello".

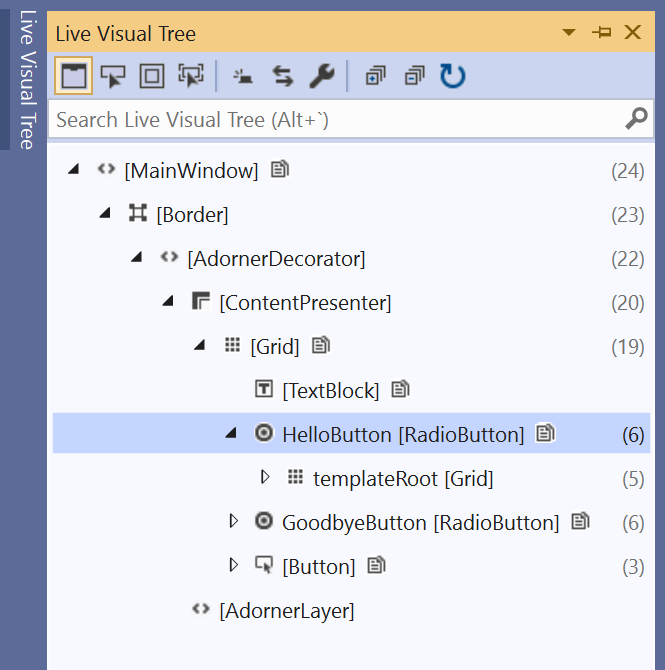
1. Нажмите кнопку **ОК** в окне сообщения, чтобы закрыть его.
2. В окне **Greetings** выберите переключатель **Goodbye** и нажмите кнопку **Display** .

Строка MessageBox.Show("Goodbye.") выделяется желтым цветом.

1. Нажмите клавишу **F5**, чтобы продолжить отладку. Когда появится окно сообщения, нажмите в нем кнопку **ОК** , чтобы закрыть его.
2. Закройте окно приложения, чтобы остановить отладку.
3. В строке меню выберите **Отладка** > **Выключить все точки останова**.

### Просмотр представления элементов пользовательского интерфейса

В работающем приложении в верхней части окна должно отображаться мини-приложение. Мини-приложение — это вспомогательное средство времени выполнения, обеспечивающее быстрый доступ к некоторым полезным возможностям отладки. Нажмите первую кнопку, **Перейти к динамическому визуальному дереву**. Должно отобразиться окно с деревом, содержащим все визуальные элементы страницы. Разверните узлы, чтобы найти добавленные кнопки.



### Сборка окончательной версии приложения

Теперь, когда вы проверили, что все работает, можно подготовить окончательную сборку приложения.

1. Выберите в главном меню **Сборка** > **Очистить решение** для удаления промежуточных файлов и выходных файлов, которые были созданы в ходе предыдущих сборок. Этот шаг не является обязательным, но позволяет очистить результаты отладочной сборки.
2. Измените конфигурацию сборки для HelloWPFApp с **Отладка** на **Выпуск** с помощью раскрывающегося списка на панели инструментов (сейчас это "Отладка").
3. Выполните сборку решения, выбрав **Сборка** > **Собрать решение**.