Бесплатная электронная книга

учусь

uwp

Free unaffiliated eBook created from
Stack Overflow contributors.

#uwp

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: uwp

It is an unofficial and free uwp ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official uwp.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с uwp

замечания

В этом разделе представлен обзор того, что такое uwp, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках uwp и ссылаться на связанные темы. Поскольку документация для uwp является новой, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Подробные инструкции по настройке или установке UWP.

Требования

1. Windows 10
2. Visual Studio 2015

меры

- Загрузите и установите пользовательскую установку Visual Studio 2015, при этом убедитесь, что `Universal Windows App Development Tools` выбраны вместе со своими дополнительными параметрами:
  а) *Инструменты и Windows SDK*
  б) *Эмулятор для Windows Phone*

- Обязательно включите режим разработки для разработки и развертывания устройства.

- Выберите шаблон на основе языка, который вы хотите использовать:
  C # , **Visual Basic** , **C ++** или **JavaScript** .

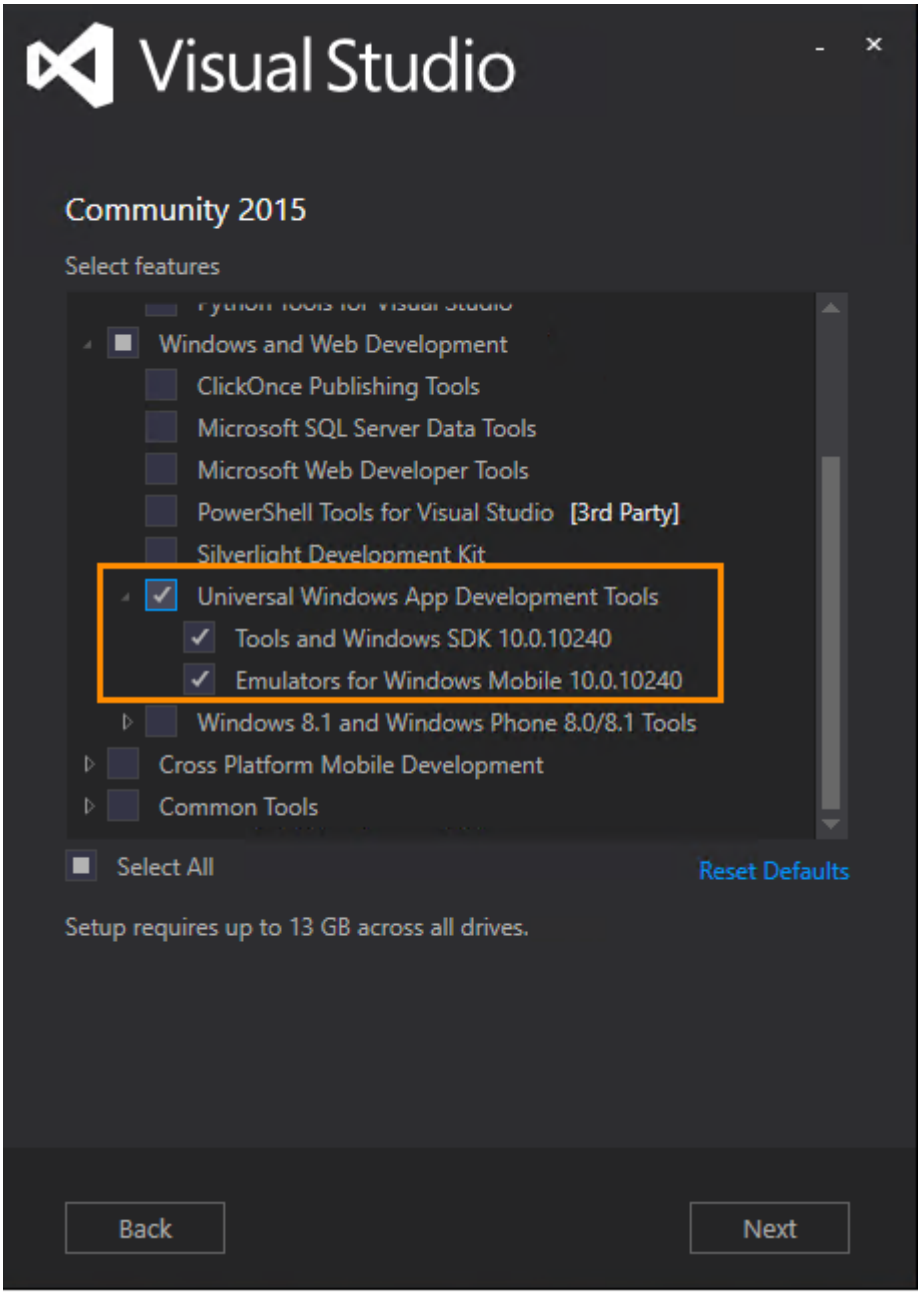
- Затем создайте пустое приложение (Universal Windows).

- Выберите целевую и минимальную версии Windows 10, подходящие для вашего приложения.

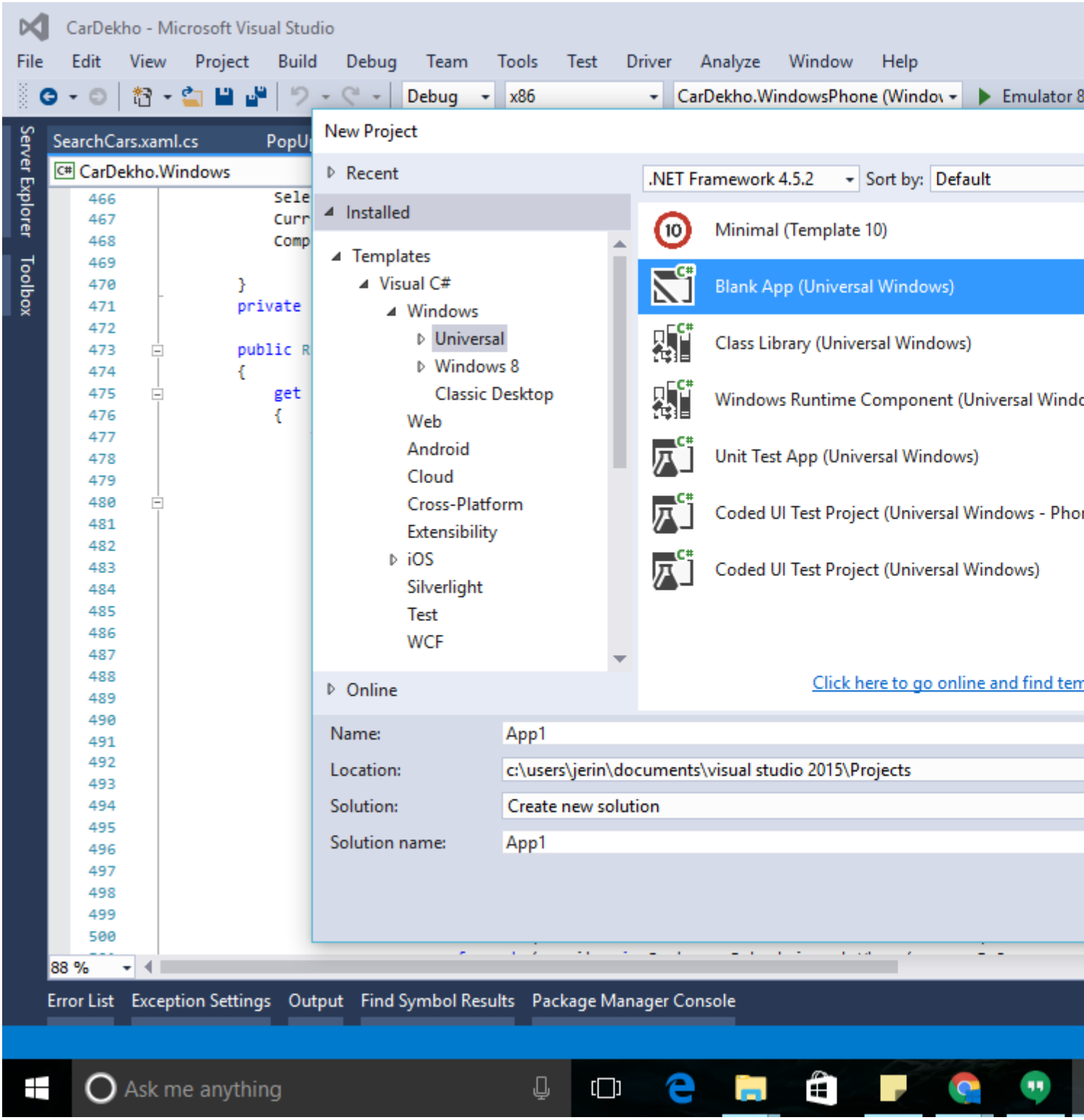
  Нажмите здесь, если вы не уверены, какие версии вы должны выбрать или просто оставите параметры по умолчанию и нажмите «OK», чтобы начать работу!

моментальные снимки

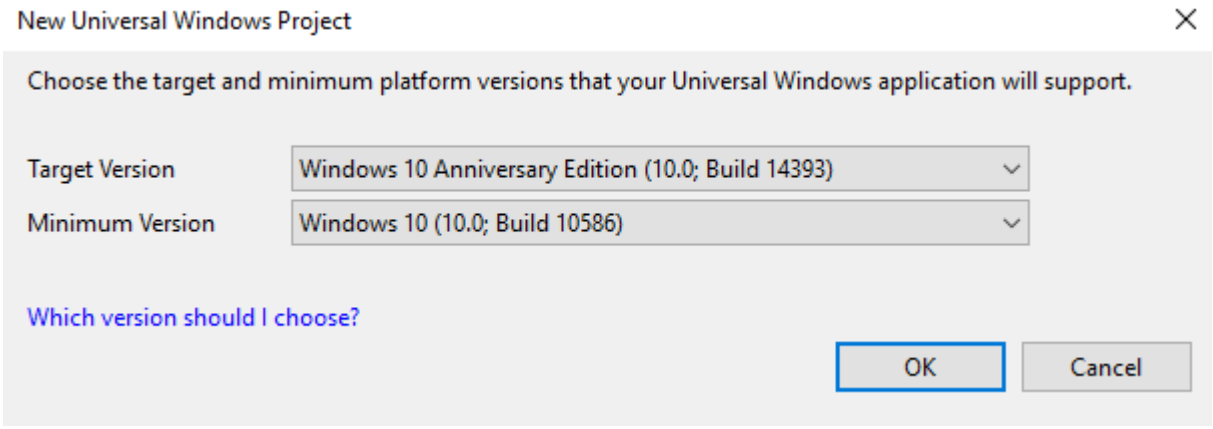
```
Installation
```




```
Creating a new project
```




```
Selecting Target and minimum version for your Application
```



Создание первого приложения UWP

В этом примере демонстрируется, как разработать простое приложение UWP.

При создании проекта «Blank App (Universal Windows)» есть много важных файлов, созданных в вашем решении.

Все файлы в вашем проекте можно увидеть в **обозревателе решений** .

Некоторые из важных файлов в вашем проекте:

- **App.xaml и App.xaml.cs** - App.xaml используется для объявления ресурсов, доступных в приложении, а App.xaml.cs - это код бэкэнд для него. App.xaml.cs является начальной точкой входа приложения
- **MainPage.xaml** - это пользовательский интерфейс запуска по умолчанию для вашего приложения (вы также можете изменить стартовую страницу приложения в App.xaml.cs)
- **Package.appxmanifest** - этот файл содержит важную информацию о вашем приложении, такую как отображаемое имя, точка входа, визуальные активы, список возможностей, информация о упаковке и т. Д.

Начиная

- **Добавление кнопки на страницу**

  Чтобы добавить элемент или инструмент UI на вашу страницу, просто перетащите элемент из окна панели инструментов слева. Найдите инструмент «Кнопка» на панели инструментов и отпустите его на странице своего приложения.

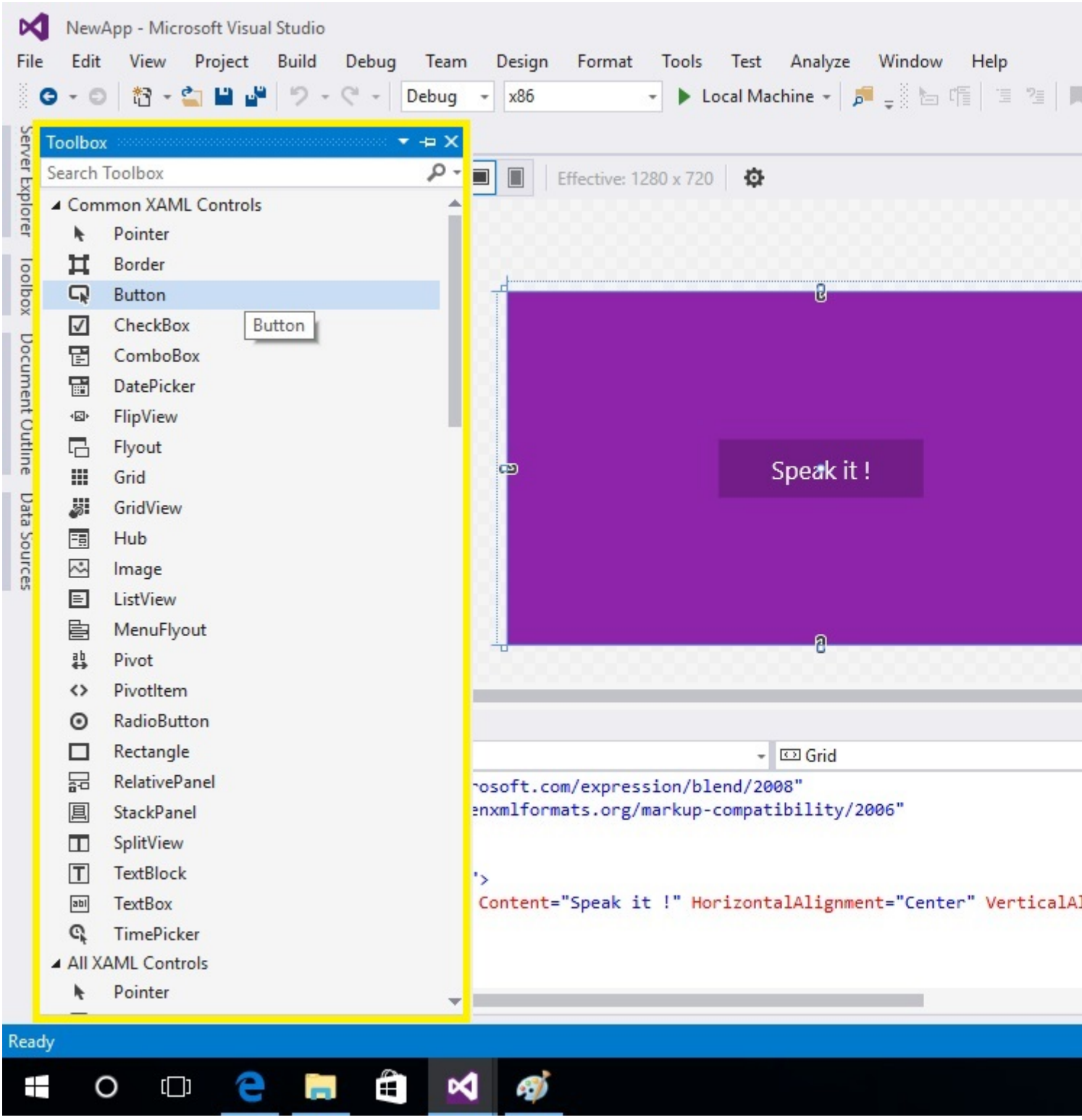
- **Настройка пользовательского интерфейса**

  Все свойства для определенного инструмента отображаются в окне свойств на нижней стороне справа.

Здесь мы изменим текст внутри кнопки «Говорить!». Для этого сначала нажмите на кнопку, чтобы выбрать ее, а затем прокрутите окно свойств, чтобы найти **контент** и изменить текст на нужную строку («Speak it!»).

Мы также изменим цвет фона для страницы. Каждая страница имеет родительский элемент (обычно это сетка), который содержит все остальные элементы. Таким образом, мы изменим цвет родительской сетки. Для этого нажмите на сетку и измените **кисть> Фон** в окне свойств на нужный вам цвет.

Пользовательский интерфейс будет выглядеть примерно так, как только вы его настроили.



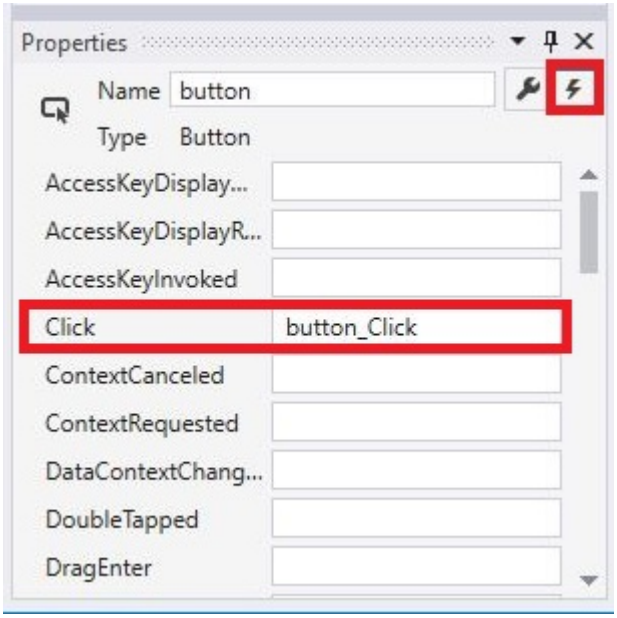
- **Код позади**

  Теперь давайте сделаем что-нибудь щелкнув по нашей кнопке!

  Нажатие на кнопку вызывает событие, и нам нужно обработать событие, чтобы сделать что-то полезное при нажатии кнопки.

  Добавление обработчика событий

Чтобы добавить обработчик события клика к вашей кнопке, выберите кнопку, перейдите в окно свойств и выберите ***значок молнии*** . Это окно состоит из всех событий, которые доступны для выбранного элемента (кнопка в нашем случае). Затем дважды щелкните текстовое поле рядом с событием *«Click»,* чтобы автоматически генерировать обработчик для события нажатия кнопки.



После этого вы будете перенаправлены на страницу ac # (MainPage.xaml.cs). Добавьте следующий код к методу обработчика событий:

```
 MediaElement mediaElement = new MediaElement();
        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();
        Windows.Media.SpeechSynthesis.SpeechSynthesisStream stream = await
synth.SynthesizeTextToStreamAsync("Hello, World!");
        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
```

Затем добавьте ключевое слово **async** в обработчик событий.

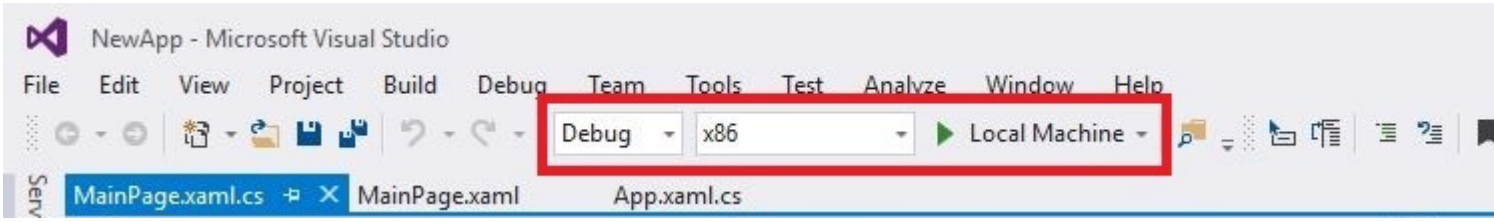
После добавления кода выше ваш класс должен выглядеть примерно так:

```
public sealed partial class MainPage : Page
{
    string speakIt = "Hello, World!";
    public MainPage()
    {
        this.InitializeComponent();
    }

    private async void button_Click(object sender, RoutedEventArgs e)
    {
        MediaElement mediaElement = new MediaElement();
        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();
        Windows.Media.SpeechSynthesis.SpeechSynthesisStream stream = await
synth.SynthesizeTextToStreamAsync(speakIt);
        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
```

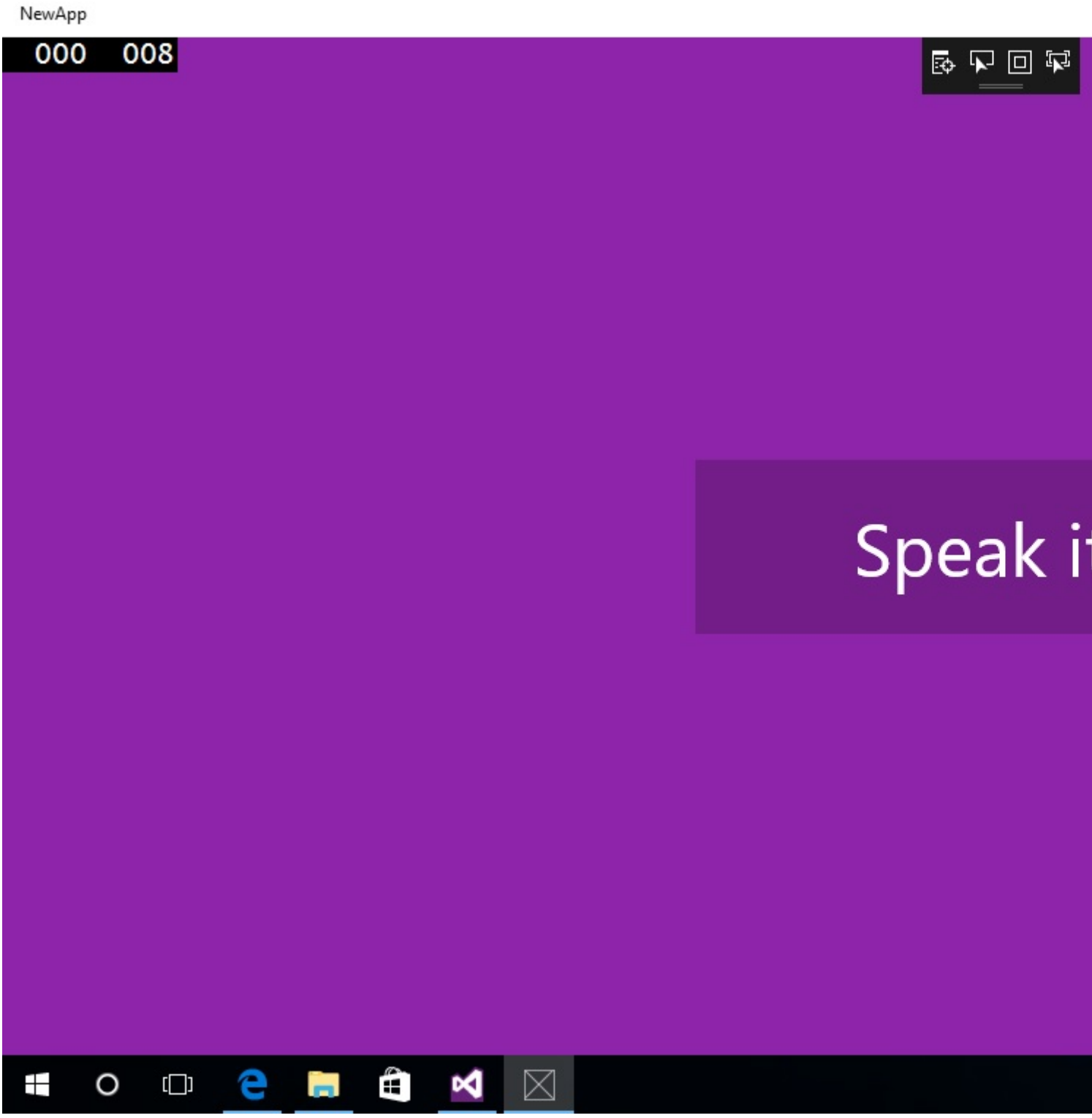
```
    }
}
```

- **Запустите приложение!**

  Ваше приложение готово к запуску. Вы можете запустить приложение, нажав F5 или выбрать свое устройство, на котором вы хотите развернуть и отладить ваше приложение, и нажать кнопку «Пуск».



После создания ваше приложение будет развернуто на вашем устройстве. В зависимости от разрешения вашего устройства и размера экрана приложение автоматически настроит свой макет. *(Вы можете изменить размер окна, чтобы увидеть, насколько он работает)*



- **Идти дальше**

  Теперь, когда вы сделали свое первое приложение, давайте сделаем еще один шаг!

  Добавьте текстовое поле на свою страницу и нажав кнопку, приложение опубликует все, что написано в текстовом поле.

  Начните с перетаскивания элемента управления TextBox из панели инструментов в макет. Затем укажите имя своего TextBox в меню свойств. *(почему нам нужно указывать имя? так что мы можем легко использовать этот элемент управления)*

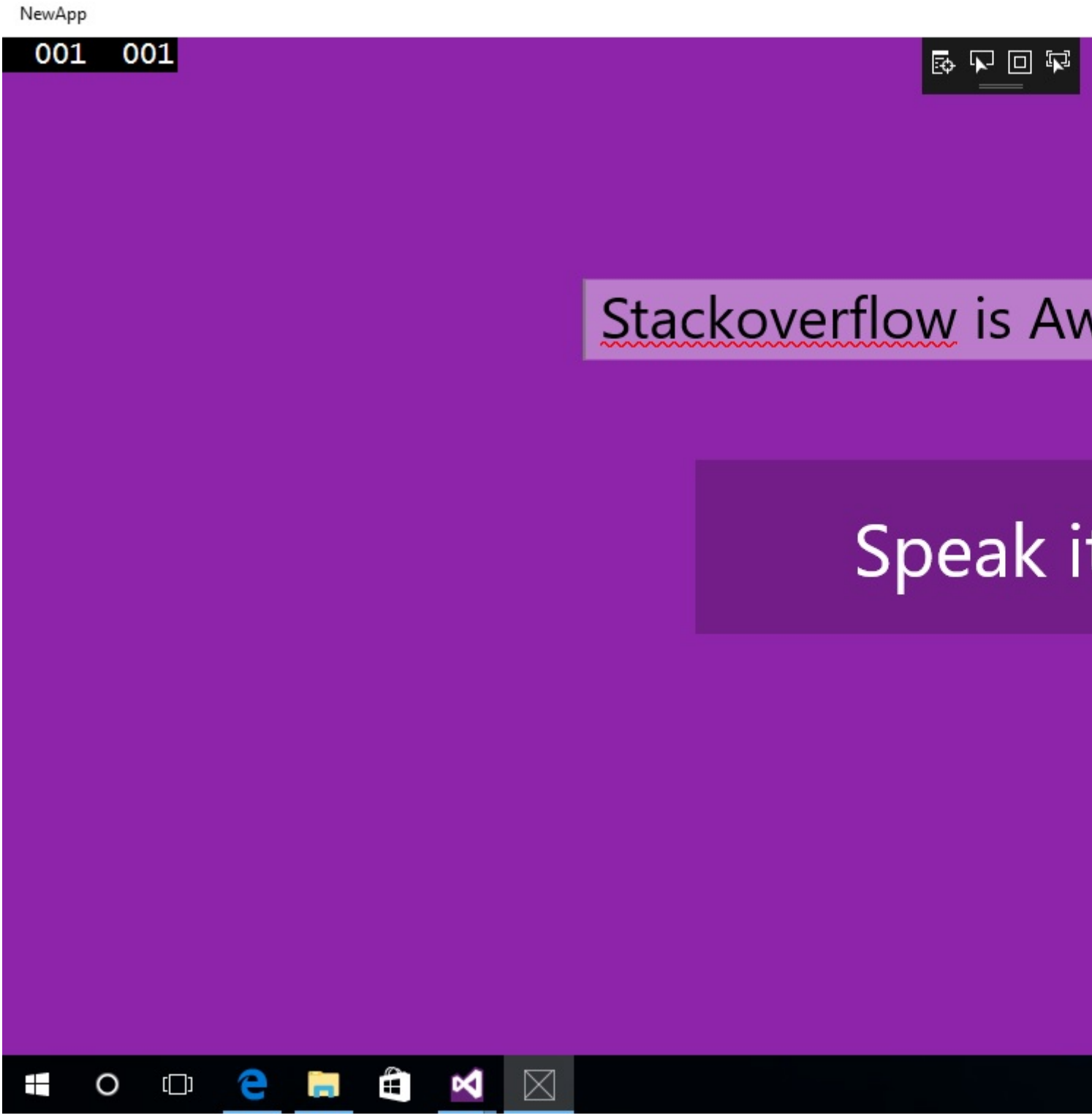
Visual Studio по умолчанию дает вашему управлению имя, но это хорошая привычка называть элементы управления в соответствии с тем, что они делают или что-то важное.

Я называю свой текстовый ящик - « *speakText* ».

```
    private async void button_Click(object sender, RoutedEventArgs e)
    {
        //checking if the text provided in the textbox is null or whitespace
        if (!string.IsNullOrWhiteSpace(speakText.Text))
            speakIt = speakText.Text;
        else
            speakIt = "Please enter a valid string!";

        MediaElement mediaElement = new MediaElement();
        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();
        Windows.Media.SpeechSynthesis.SpeechSynthesisStream stream = await
synth.SynthesizeTextToStreamAsync(speakIt);
        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();

    }
```

Теперь разверните свой код!

Теперь ваше приложение может выдать любую *допустимую* строку, которую вы ему предоставляете!



Поздравляем! Вы успешно создали свое собственное приложение UWP!

Прочитайте Начало работы с uwp онлайн: https://riptutorial.com/ru/uwp/topic/1069/начало-работы-с-uwp

глава 2: UWP Hello World

Синтаксис

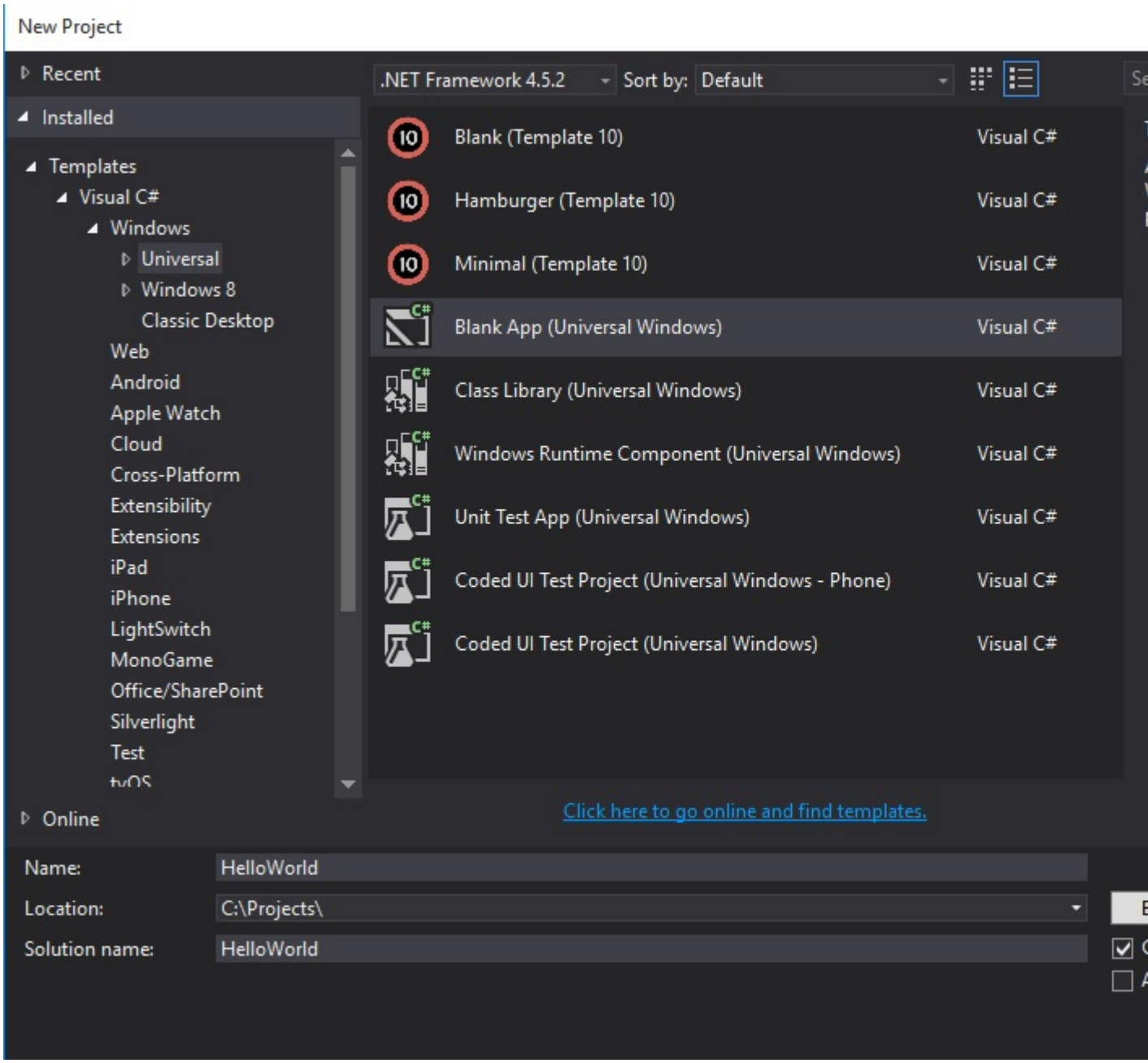
- Это простой пример популярного «Hello World!». для универсальной платформы Windows в Windows 10.

Examples

Hello World - универсальная платформа Windows

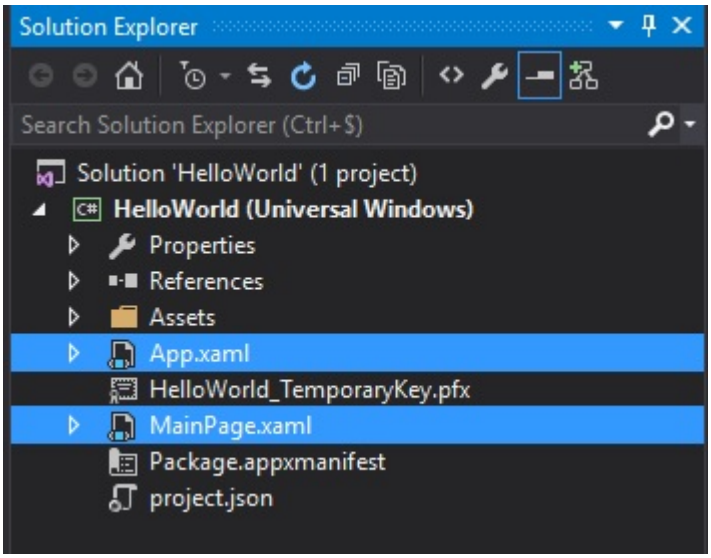
После запуска Visual Studio 2015 перейдите в `File → New → Project` . В диалоговом окне « *Новый проект* » перейдите в дерево шаблонов к `Visual C# → Windows → Universal` и выберите « `Blank App (Universal Windows)` . Затем нам нужно заполнить форму для описания приложения:

1. **Имя** : это имя приложения, которое будет отображаться пользователю. Установите его в `HelloWorld` или используйте пользовательский заголовок.
2. **Местоположение** : указывает, где будет храниться проект
3. **Название решения** : это своего рода контейнер проектов, который объединяет несколько проектов, связанных с одним и тем же приложением (например, решение может состоять из проекта пользовательского интерфейса и модельного проекта). Вы можете указать то же `Name` что и ваш первоначальный проект.



Содержимое проекта по умолчанию

Вы получите проект со следующими файлами:



1. **Package.appxmanifest** : описывает свойства вашего приложения. Он содержит некоторые настройки пользовательского интерфейса, такие как его имя disaply, его логотип, поддерживаемые вращения. В нем также содержатся технические параметры, такие как точка входа приложения (по умолчанию это класс `App`). Наконец, он также перечисляет авторизации, которые требуются вашему приложению на вкладке « *Возможности* »; например, если вы хотите использовать веб-камеру в своем приложении, вам нужно будет проверить соответствующие возможности.
2. **App.xaml / App.xaml.cs** : класс `App` является точкой входа по умолчанию вашего приложения. Файлы xaml могут содержать ресурсы, общие для всего приложения, такие как настройки стилей или экземпляр класса, который вы хотите использовать, например локатор ViewModel. Файлы, содержащие код, содержат весь код запуска приложения. По умолчанию он реализует метод `OnLaunched` который вызывается конечным пользователем. Он инициализирует окно и перемещается на первую страницу приложения (по умолчанию класс `MainPage`).
3. **MainPage.xaml / MainPage.xaml.cs** : это начальная страница нашего приложения. Он содержит только пустую сетку, которая является элементом управления компоновкой.

Изменить вид

Откройте `MainPage.xaml` и замените элемент управления сеткой на

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Click="Button_Click">Say Hello !</Button>
    <TextBlock Grid.Column="1"
               VerticalAlignment="Center"
               x:Name="myText"
```

```
                Text="Click the button." />
</Grid>
```

Это создаст сетку с двумя столбцами. Первый столбец в качестве ширины, заданный `auto` что означает, что он будет автоматически установлен в зависимости от размера его дочерних элементов. Второй столбец будет растянут, чтобы заполнить оставшееся пространство в окне. Эта сетка содержит два элемента:

- `Button` которая находится в первом столбце. Событие клика привязывается к методу `Button_Click` в коде, а его текст под заголовком *«Скажите привет!».* ,
- `TextBlock` который находится во втором столбце. В этом тексте установлен *«Нажмите кнопку».* , И мы установили имя для этого элемента управления с помощью атрибута `x:Name` . Это необходимо для использования элемента управления в коде. В `MainPage.xaml.cs` добавьте следующий код:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    this.myText.Text = "Hello World!";
}
```

Это метод, который будет вызываться, когда пользователь щелкает (или отжимает) кнопку. И он обновит `TextBlock` и установит его текст в *«Hello World!».* ,

Запуск приложения

Чтобы запустить приложение, вы можете использовать меню `Debug → Start Debugging` или ярлык `F5` . По умолчанию он запустит приложение на вашем локальном компьютере.

Прочитайте UWP Hello World онлайн: https://riptutorial.com/ru/uwp/topic/2339/uwp-hello-world

глава 3: WebView

Examples

Добавить веб-интерфейс в пользовательский интерфейс

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <WebView x:Name="MyWebView" />
</Grid>
```

Открыть веб-сайт

```
MyWebView.Navigate(new Uri("http://www.url.com"));
```

Открыть локальную страницу html

```
MyWebView.Navigate(new Uri("ms-appdata:///local/Downloads/index.html"));
```

Прочитайте WebView онлайн: https://riptutorial.com/ru/uwp/topic/6541/webview

глава 4: Адаптивный пользовательский интерфейс

Examples

Используйте AdaptiveTrigger для изменения макета пользовательского интерфейса

Приложения UWP могут работать в оконном режиме и на нескольких устройствах. Они могут отображаться на широком диапазоне размеров экрана от телефонов с низким уровнем громкости до огромного экрана концентратора. Использование относительного позиционирования будет достаточно для большого количества сценариев, но по мере увеличения размера окна всегда интересно полностью изменить макет, перемещая элементы управления страницы в разные местоположения.

В этом примере мы будем использовать вертикальную компоновку на узких экранах и горизонтальную компоновку на широком экране. На огромных широких экранах мы также изменим размеры элементов.

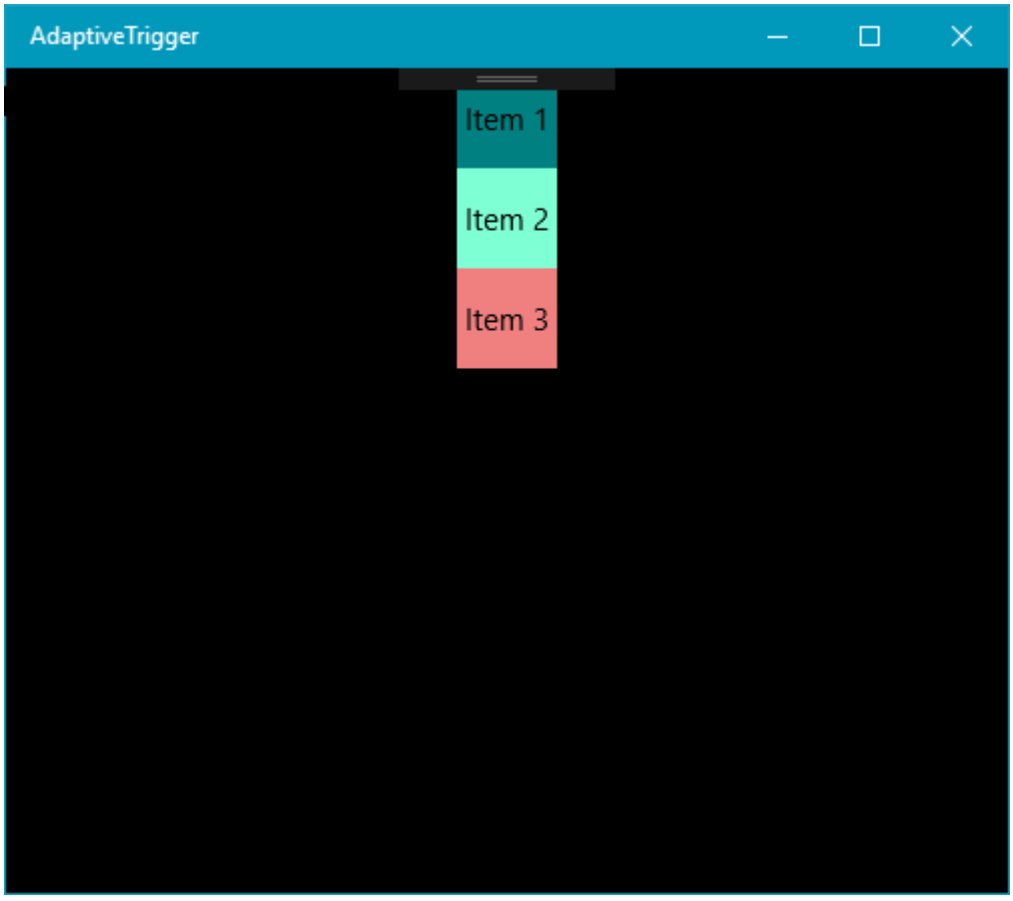
```
<Border x:Name="item2"
        Background="Aquamarine"
        Width="50"
        Height="50">
    <TextBlock Text="Item 2"
               VerticalAlignment="Center"
               HorizontalAlignment="Center" />
</Border>

<Border x:Name="item3"
        Background="LightCoral"
        Width="50"
        Height="50">
    <TextBlock Text="Item 3"
               VerticalAlignment="Center"
               HorizontalAlignment="Center" />
</Border>

<VisualStateManager.VisualStateGroups>
    <VisualStateGroup>

        <VisualState x:Name="ultrawide">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="800" />
            </VisualState.StateTriggers>

            <VisualState.Setters>
                <Setter Target="mainPanel.Orientation" Value="Horizontal" />
                <Setter Target="item1.Width" Value="100" />
                <Setter Target="item1.Height" Value="100" />
                <Setter Target="item2.Width" Value="100" />
```

```
                <Setter Target="item2.Height" Value="100" />
                <Setter Target="item3.Width" Value="100" />
                <Setter Target="item3.Height" Value="100" />
            </VisualState.Setters>
        </VisualState>

        <VisualState x:Name="wide">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="600" />
            </VisualState.StateTriggers>

            <VisualState.Setters>
                <Setter Target="mainPanel.Orientation" Value="Horizontal" />
            </VisualState.Setters>
        </VisualState>

        <VisualState x:Name="narrow" />
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

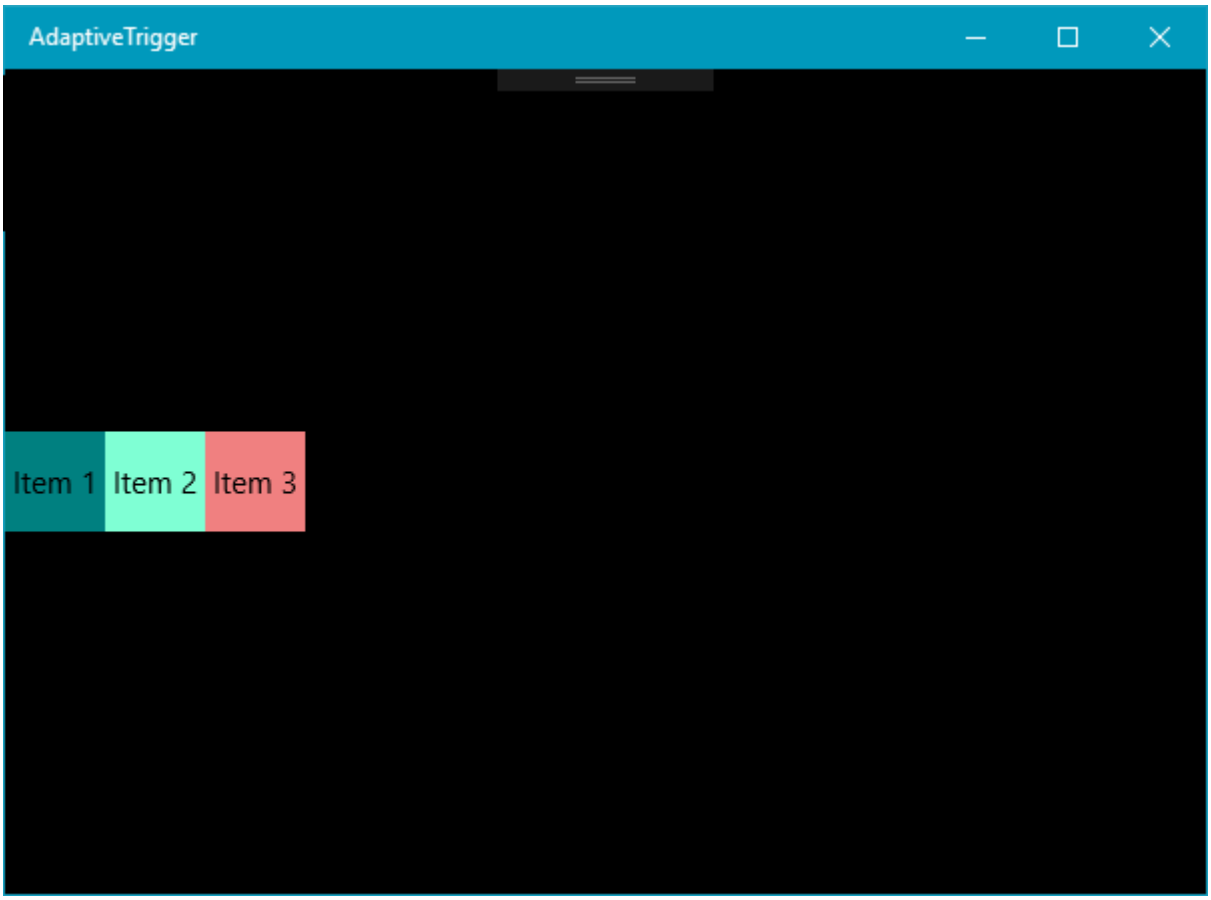
Когда окно будет изменено, система сравнит ширину текущего окна с минимальной шириной от AdaptiveTrigger. Если текущая ширина больше или равна минимальной ширине, триггер активируется и отображается соответствующая VisualState.

Вот результат для разных состояний

узкий



Широкий



ультраширокую

Прочитайте Адаптивный пользовательский интерфейс онлайн:
https://riptutorial.com/ru/uwp/topic/6420/адаптивный-пользовательский-интерфейс

глава 5: Жизненный цикл приложения

Вступление

Универсальный жизненный цикл Windows 10 App состоит из трех разных состояний: 1) Запуск - приложение используется в настоящее время 2) Не работает - приложение закрыто и удалено из памяти 3) Suspended - состояние приложения заморожено, но оно все еще в памяти [! [введите описание изображения здесь] [1]] [1] [1]: https://i.stack.imgur.com/x7MCl.png Как вы можете видеть на картинке выше, существуют различные события, связанные с переходом из одного состояния в другой. В разделе примеров я покажу, как их обрабатывать.

замечания

Хорошо ссылаться на две хорошие статьи в блоге MSDN:

1. https://msdn.microsoft.com/en-us/windows/uwp/launch-resume/app-lifecycle
2. https://blogs.windows.com/buildingapps/2016/04/28/the-lifecycle-of-a-uwp-app/#RqKAKkevsAPIvBUT.97

Examples

Управление «запуском»

При переходе в состояние «Запуск» есть специальный обработчик, связанный с этим событием: Откройте класс «App.xaml.cx» и см. Метод «OnLaunched» - он активируется, когда приложение открывается пользователем из состояния «Terminaded»:

```
protected override void OnLaunched(LaunchActivatedEventArgs e)
    {
        Frame rootFrame = Window.Current.Content as Frame;

        // Do not repeat app initialization when the Window already has content,
        // just ensure that the window is active
        if (rootFrame == null)
        {
            // Create a Frame to act as the navigation context and navigate to the first page
            rootFrame = new Frame();

            rootFrame.NavigationFailed += OnNavigationFailed;

           //You can get information about previous state of the app:

            if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)
            {
                //The app was previously suspended but was then shutdown
                //at some point because the system needed to reclaim memory.
```

```
            }
            if (e.PreviousExecutionState == ApplicationExecutionState.ClosedByUser)
            {
                //The user closed the app with the close gesture in tablet mode,
                //or with Alt+F4.When the user closes the app, it is first suspended
                //and then terminated.
            }
            if (e.PreviousExecutionState == ApplicationExecutionState.NotRunning)
            {
                //An app could be in this state because it hasn't been launched since the last
time
                //the user rebooted or logged in. It can also be in this state if it was
running
                //but then crashed, or because the user closed it earlier.
            }
            if (e.PreviousExecutionState == ApplicationExecutionState.Running)
            {
                //The app was already open when the user tried to launch it again
            }
            if (e.PreviousExecutionState == ApplicationExecutionState.Suspended)
            {
                //The user either minimized or switched away from your app
                //and didn't return to it within a few seconds.
            }

            // Place the frame in the current Window
            Window.Current.Content = rootFrame;
        }

        //When available system resources allow, the startup performance of Windows Store
        //apps on desktop device family devices is improved by proactively launching
        //the user’s most frequently used apps in the background. A prelaunched app
        //is put into the suspended state shortly after it is launched.Then, when the
        //user invokes the app, the app is resumed by bringing it from the suspended
        //state to the running state--which is faster than launching the app cold.
        //The user's experience is that the app simply launched very quickly.
        if (e.PrelaunchActivated == false)
        {
            if (rootFrame.Content == null)
            {
                rootFrame.Navigate(typeof(MainPage), e.Arguments);
            }
            Window.Current.Activate();
        }
    }
```

«Приостановка» обработки состояния

При переходе в состояние «Приостановлено» есть специальный обработчик, связанный с этим событием: Откройте «App.xaml.cx» класс и см. Конструктор «App» - есть обработчик событий:

```
public App()
    {
        this.InitializeComponent();
        //Handle suspending operation with event handler:
        this.Suspending += OnSuspending;
    }
```

Теперь вы можете обработать событие приостановки:

```
private Dictionary<string, object> _store = new Dictionary<string, object>();
private readonly string _saveFileName = "store.xml";
private async void OnSuspending(object sender, SuspendingEventArgs e)
    {
        var deferral = e.SuspendingOperation.GetDeferral();
        _store.Add("timestamp", DateTime.Now);
        await SaveStateAsync();
        //TODO: Save application state and stop any background activity
        //Here you can use  await SuspensionManager.SaveAsync();
        //To read more about saving state please refer to below MSDN Blog article:
        //https://blogs.windows.com/buildingapps/2016/04/28/the-lifecycle-of-a-uwp-
app/#RqKAKkevsAPIvBUT.97
        deferral.Complete();
    }

    private async Task SaveStateAsync()
    {
        var ms = new MemoryStream();
        var serializer = new DataContractSerializer(typeof(Dictionary<string, object>));
        serializer.WriteObject(ms, _store);

        var file = await ApplicationData.Current.LocalFolder.CreateFileAsync(_saveFileName,
CreationCollisionOption.ReplaceExisting);

        using (var fs = await file.OpenStreamForWriteAsync())
        {
            //because we have written to the stream, set the position back to start
            ms.Seek(0, SeekOrigin.Begin);
            await ms.CopyToAsync(fs);
            await fs.FlushAsync();
        }
    }
```

«Возобновление» обработки состояния

Ваше приложение может быть открыто пользователем из состояния «Приостановлено». При выполнении этого используется обработчик событий «OnResuming». В классе « App.xaml.cs»:

```
public App()
    {
        this.InitializeComponent();
        this.Suspending += OnSuspending;
        //Handle resuming operation:
        this.Resuming += App_Resuming;
    }

private void App_Resuming(object sender, object e)
    {
        //Do some operation connected with app resuming for instance refresh data
    }
```

Прочитайте Жизненный цикл приложения онлайн: https://riptutorial.com/ru/uwp/topic/8135/жизненный-цикл-приложения

глава 6: Задачи UWP-фона

замечания

- Для регистрации фоновой задачи, которая выполняется в отдельном процессе, вам нужно перейти на вкладку «Объявления» в Package.appxmanifest и добавить новую «Фоновая задача» и установить точку входа.
- Регистрация фоновой задачи одного процесса может быть выполнена с помощью `BackgroundTaskBuilder` , но приложение будет генерировать исключение, если вы дважды зарегистрируете задачу, поэтому вы должны проверить, была ли вы уже зарегистрирована задача.
- Приложение должно получить полномочия для регистрации новой задачи, это можно сделать, вызвав `BackgroundExecutionManager.RequestAccessAsync()` , но убедитесь, что у вас действительно есть разрешение. Вызов возвращает тип доступа (`BackgroundAccessStatus` enum), который укажет, есть ли у вас доступ или нет.
- Записанные задачи сохраняются до тех пор, пока пакет не будет удален, но не повредит для проверки задач, которые вам нужны при каждом запуске, ошибка возникает!
- Когда приложение обновляется, разрешение на регистрацию новой задачи отменяется. Чтобы ваше приложение запускалось после обновления, особенно если вы добавили новый регистр задач, вам необходимо *удалить и запросить доступ* с помощью `BackgroundAccessManager` . Один из способов узнать, обновляется ли ваше приложение, - зарегистрировать другую задачу с помощью `SystemTrigger` , типа `SystemTriggerType.ServicingComplete` .

Examples

Регистрация задачи

```
/// <summary>
/// Registers a background task in the system waiting to trigger
/// </summary>
/// <param name="taskName">Name of the task. Has to be unique</param>
/// <param name="taskEntryPoint">Entry point (Namespace) of the class (has to implement
IBackgroundTask and has to be in a Windows Runtime Component) to start</param>
/// <param name="trigger">What has to be triggered to start the task</param>
/// <param name="condition">Optional condition. Can be null</param>
/// <param name="recreateIfExists">Should the Task be recreated if it already exists?</param>
/// <returns></returns>
public BackgroundTaskRegistration RegisterTask(string taskName, string taskEntryPoint,
IBackgroundTrigger trigger, IBackgroundCondition condition = null) {
    Debug.WriteLine("Try registering task: " + taskName);

    var builder = new BackgroundTaskBuilder {
        Name = taskName,
        TaskEntryPoint = taskEntryPoint
```

```
    };

    builder.SetTrigger(trigger);

    if (condition != null) {
        builder.AddCondition(condition);
    }

    try {
        var task = builder.Register();
        Debug.WriteLine("Task successfully registered");
        return task;
    } catch (Exception exception) {
        Debug.WriteLine("Error creating Task: " + exception);
        return null;
    }
}
```

Получить зарегистрированную задачу по ее названию

```
/// <summary>
/// Gets a BackgroundTask by its name
/// </summary>
/// <param name="taskName">Name of the task to find</param>
/// <returns>The found Task or null if none found</returns>
public BackgroundTaskRegistration TaskByName(string taskName) =>
       BackgroundTaskRegistration.AllTasks.FirstOrDefault(x =>
x.Value.Name.Equals(taskName)).Value as BackgroundTaskRegistration;
```

Задание

```
public sealed class BackgroundTask : IBackgroundTask {

    private BackgroundTaskDeferral _deferral;

    /// <summary>
    /// Registers the listener to check if the button is pressed
    /// </summary>
    /// <param name="taskInstance">An interface to an instance of the background task. The
system creates this instance when the task has been triggered to run.</param>
    public async void Run(IBackgroundTaskInstance taskInstance) {
        _deferral = taskInstance.GetDeferral();

        //Do async operations here

        _deferral.Complete();
    }
}
```

Проверьте, зарегистрирована ли задача

```
private bool IsTaskRegistered(string taskName) =>
        BackgroundTaskRegistration.AllTasks.Any(x => x.Value.Name.Equals(taskName));
```

Запуск задачи вручную

```
var trigger = new ApplicationTrigger();
TaskHandlerMentionedInThisTutorial.RegisterTask(TaskName, entryPoint, trigger, null, true);

await trigger.RequestAsync();
```

Отмена регистрации задачи

```
/// <summary>
/// Unregister a single background task with given name
/// </summary>
/// <param name="taskName">task name</param>
/// <param name="cancelTask">true if task should be cancelled, false if allowed to
finish</param>
public void UnregisterTask(string taskName, bool cancelTask) =>
    BackgroundTaskRegistration.AllTasks.First(x =>
x.Value.Name.Equals(taskName)).Value?.Unregister(cancelTask);

/// <summary>
/// Unregister an active group of background tasks, which name contains given string
/// </summary>
/// <param name="taskNamePart">part of the task name</param>
/// <param name="cancelTask">true if tasks should be cancelled, false if allowed to
finish</param>
public void UnregisterTasks(string taskNamePart, bool cancelTask)
{
    foreach (var task in BackgroundTaskRegistration.AllTasks.Where(x =>
x.Value.Name.Contains(taskNamePart)))
        task.Value.Unregister(cancelTask);
}
```

Регистрация фоновой задачи с помощью триггера

Фоновая задача - отличный способ выполнить некоторую работу, пока ваше приложение не запущено. Прежде чем вы сможете использовать их, вам придется их зарегистрировать.

Вот пример базового класса задачи, включающий регистрацию с триггером и условием, и реализацию Run

```
public sealed class Agent : IBackgroundTask
{
    public void Run(IBackgroundTaskInstance taskInstance)
    {
        // run the background task code
    }

    // call it when your application will start.
    // it will register the task if not already done
    private static IBackgroundTaskRegistration Register()
    {
        // get the entry point of the task. I'm reusing this as the task name in order to get
an unique name
        var taskEntryPoint   = typeof(Agent).FullName;
```

```
        var taskName         = taskEntryPoint;

        // if the task is already registered, there is no need to register it again
        var registration            = BackgroundTaskRegistration.AllTasks.Select(x =>
x.Value).FirstOrDefault(x => x.Name == taskName);
        if(registration != null) return registration;

        // register the task to run every 30 minutes if an internet connection is available
        var taskBuilder             = new BackgroundTaskBuilder();
        taskBuilder.Name            = taskName;
        taskBuilder.TaskEntryPoint  = taskEntryPoint;
        taskBuilder.SetTrigger(new TimeTrigger(30, false));
        taskBuilder.AddCondition(new SystemCondition(SystemConditionType.InternetAvailable));

        return taskBuilder.Register();
    }
}
```

Прочитайте Задачи UWP-фона онлайн: https://riptutorial.com/ru/uwp/topic/2494/задачи-uwp-фона

глава 7: Изображений

параметры

параметр	Описание
DecodePixelWidth	Будут загружены BitmapImage с указанной шириной. Помогает с использованием памяти и скоростью при загрузке больших изображений, которые должны отображаться меньше на экране. Это более эффективно, чем загрузка полного изображения, и полагайтесь на элемент управления `Image` чтобы изменить размер.
DecodePixelHeight	То же, что `DecodePixelHeight` . Если задан только один параметр, система будет поддерживать соотношение сторон изображения при загрузке требуемого размера.

Examples

Использование BitmapImage с контролем изображения

```
<Image x:Name="MyImage" />
```

```
// Show image from web
MyImage.Source = new BitmapImage(new Uri("http://your-image-url.com"))

// Show image from solution
MyImage.Source = new Uri("ms-appx:///your-image-in-solution", UriKind.Absolute)

// Show image from file
IRandomAccessStreamReference file = GetFile();
IRandomAccessStream fileStream = await file.OpenAsync();
var image = new BitmapImage();
await image.SetSourceAsync(fileStream);
MyImage.Source = image;
fileStream.Dispose(); // Don't forget to close the stream
```

Отображение элементов управления с помощью RenderTargetBitmap

```
<TextBlock x:Name="MyControl"
           Text="Hello, world!" />
```

```
var rtb = new RenderTargetBitmap();
await rtb.RenderAsync(MyControl); // Render control to RenderTargetBitmap

// Get pixels from RTB
IBuffer pixelBuffer = await rtb.GetPixelsAsync();
```

```
byte[] pixels = pixelBuffer.ToArray();

// Support custom DPI
DisplayInformation displayInformation = DisplayInformation.GetForCurrentView();

var stream = new InMemoryRandomAccessStream();
BitmapEncoder encoder = await BitmapEncoder.CreateAsync(BitmapEncoder.PngEncoderId, stream);
encoder.SetPixelData(BitmapPixelFormat.Bgra8, // RGB with alpha
                     BitmapAlphaMode.Premultiplied,
                     (uint)rtb.PixelWidth,
                     (uint)rtb.PixelHeight,
                     displayInformation.RawDpiX,
                     displayInformation.RawDpiY,
                     pixels);

await encoder.FlushAsync(); // Write data to the stream
stream.Seek(0); // Set cursor to the beginning

// Use stream (e.g. save to file)
```

Преобразование битмапа (например, из содержимого буфера обмена) в PNG

```
IRandomAccessStreamReference bitmap = GetBitmap();
IRandomAccessStreamWithContentType stream = await bitmap.OpenReadAsync();
BitmapDecoder decoder = await BitmapDecoder.CreateAsync(stream);
var pixels = await decoder.GetPixelDataAsync();
var outStream = new InMemoryRandomAccessStream();

// Create encoder for PNG
var encoder = await BitmapEncoder.CreateAsync(BitmapEncoder.PngEncoderId, outStream);

// Get pixel data from decoder and set them for encoder
encoder.SetPixelData(decoder.BitmapPixelFormat,
                     BitmapAlphaMode.Ignore, // Alpha is not used
                     decoder.OrientedPixelWidth,
                     decoder.OrientedPixelHeight,
                     decoder.DpiX, decoder.DpiY,
                     pixels.DetachPixelData());

await encoder.FlushAsync(); // Write data to the stream

// Here you can use your stream
```

Загрузить изображение в XAML

```
<Image Source="ms-appx:///Assets/Windows_10_Hero.png"/>
```

Ваше изображение является частью приложения в папке «Активы» и помечено как «`Content`

```
<Image Source="ms-appdata:///local/Windows_10_Hero.png"/>
```

Ваше изображение было сохранено в локальной папке вашего приложения

```
<Image Source="ms-appdata:///roaming/Windows_10_Hero.png"/>
```

Ваше изображение было сохранено в папке «Роуминг» вашего приложения

Загрузить изображение из активов в коде

```
 ImageSource result = new BitmapImage(new Uri("ms-appx:///Assets/Windows_10_Hero.png"));
```

Используйте результат, чтобы установить свойство `Source` элемента управления `Image` хотя `Binding` или код

Загрузить изображение из StorageFile

```
public static async Task<ImageSource> FromStorageFile(StorageFile sf)
{
    using (var randomAccessStream = await sf.OpenAsync(FileAccessMode.Read))
    {
        var result = new BitmapImage();
        await result.SetSourceAsync(randomAccessStream);
        return result;
    }
}
```

Используйте результат, чтобы установить свойство `Source` элемента управления `Image` хотя `Binding` или код

Полезно, когда вам нужно открывать изображения, которые хранятся на диске пользователя и не поставляются с вашим приложением

Отображение элемента пользовательского интерфейса для изображения

```
public static async Task<WriteableBitmap> RenderUIElement(UIElement element)
{
    var bitmap = new RenderTargetBitmap();
    await bitmap.RenderAsync(element);
    var pixelBuffer = await bitmap.GetPixelsAsync();
    var pixels = pixelBuffer.ToArray();
    var writeableBitmap = new WriteableBitmap(bitmap.PixelWidth, bitmap.PixelHeight);
    using (Stream stream = writeableBitmap.PixelBuffer.AsStream())
    {
        await stream.WriteAsync(pixels, 0, pixels.Length);
    }
    return writeableBitmap;
}
```

Поскольку `WriteableBitmap` - это `ImageSource` вы можете использовать его для установки свойства Source элемента управления Image, хотя привязка или кодирование

Сохраните записьableBitmap в поток

```
public static async Task<IRandomAccessStream>
ConvertWriteableBitmapToRandomAccessStream(WriteableBitmap writeableBitmap)
{
    var stream = new InMemoryRandomAccessStream();

    BitmapEncoder encoder = await BitmapEncoder.CreateAsync(BitmapEncoder.JpegEncoderId,
stream);
    Stream pixelStream = writeableBitmap.PixelBuffer.AsStream();
    byte[] pixels = new byte[pixelStream.Length];
    await pixelStream.ReadAsync(pixels, 0, pixels.Length);

    encoder.SetPixelData(BitmapPixelFormat.Bgra8, BitmapAlphaMode.Ignore,
(uint)writeableBitmap.PixelWidth, (uint)writeableBitmap.PixelHeight, 96.0, 96.0, pixels);
    await encoder.FlushAsync();

    return stream;
}
```

Используйте поток для сохранения Bitmap в файл.

Прочитайте Изображений онлайн: https://riptutorial.com/ru/uwp/topic/5170/изображений

глава 8: Изображений

Examples

Присвоение BitmapImage источнику изображения

```
Image img = new Image();
BitmapImage bitmap = new BitmapImage(new Uri("ms-appx:///Path-to-image-in-solution-directory",
UriKind.Absolute));
img.Source = bitmap;
```

Прочитайте Изображений онлайн: https://riptutorial.com/ru/uwp/topic/6564/изображений

глава 9: Использование JavaScript в WebView

Вступление

В этом документе показано, как вы можете использовать JavaScript в WebView.

Этот документ охватывает: Получение HTML из WebView, ввод текста в текстовое поле на веб-сайте, имитация щелчка, чтобы нажать кнопку веб-сайта

Синтаксис

- `await webView.InvokeScriptAsync("eval", new string[] { functionString })` - для использования JavaScript
- `.documentElement` - получить ссылку на корневой узел документа
- `.getElementsByClassName(Class_Name)` - для получения элементов usign Имя класса
- `.getElementsByTagName(Tab_Name)` - для получения элементов с использованием имени тега
- `.getElementById(ID)` - для получения элемента с использованием идентификатора
- `.nodeName` - получить имя узла
- `.childNodes` - для получения дочерних элементов
- `.outerHTML` - Получить внешний HTML- `.outerHTML`
- `.innerHTML` - Получить внутренний HTML
- `.innerText` - Получить или установить InnerText
- `.click()` - для имитации щелчка

замечания

Вот пример приложения LogIn для StackOverFlow

Examples

Получение HTML из WebView

Используйте `.outerHTML` чтобы получить HTML

Вот пример кода, чтобы получить весь HTML-сайт

```
private async void GetHTMLAsync()
{
    var siteHtML = await webView.InvokeScriptAsync("eval", new string[] {
```

```
"document.documentElement.outerHTML;" });
}
```

Ввод текста в текстовое поле на веб-сайте

Используйте `.innerText` чтобы установить значение

Вот пример кода для ввода текста в поле поиска на веб-сайте Bing

```
private async void EnterTextAsync(string enterText)
{
    var functionString =
string.Format(@"document.getElementsByClassName('b_searchbox')[0].innerText = '{0}';",
enterText);
    await webView.InvokeScriptAsync("eval", new string[] { functionString });
}
```

Имитировать клик, чтобы щелкнуть кнопку веб-сайта

Используйте `.click()` для имитации щелчка

Вот пример кода, чтобы нажать кнопку поиска на веб-сайте Bing

```
private async void SimulateClickAsync()
{
    var functionString =
string.Format(@"document.getElementsByClassName('b_searchboxSubmit')[0].click();");
    await webView.InvokeScriptAsync("eval", new string[] { functionString });
}
```

Прочитайте Использование JavaScript в WebView онлайн:
https://riptutorial.com/ru/uwp/topic/10794/использование-javascript-в-webview

глава 10: Как получить текущую DateTime в C ++ UWP

Вступление

В документации для `DateTime::UniversalTime` указано:

«64-битное целое число со знаком, которое представляет собой момент времени, равный числу 100-наносекундных интервалов до или после полуночи 1 января 1601 года (согласно Григорианскому календарю)».

Это то же самое, что и структура Win32 `FILETIME` которую нужно преобразовать в 100-наносекундное длинное значение и установить его в поле `DateTime::UniversalTime` .

Examples

GetCurrentDateTime ()

```
#include <windows.h>

static Windows::Foundation::DateTime GetCurrentDateTime() {
    // Get the current system time
    SYSTEMTIME st;
    GetSystemTime(&st);

    // Convert it to something DateTime will understand
    FILETIME ft;
    SystemTimeToFileTime(&st, &ft);

    // Conversion to DateTime's long long is done vie ULARGE_INTEGER
    ULARGE_INTEGER ui;
    ui.LowPart = ft.dwLowDateTime;
    ui.HighPart = ft.dwHighDateTime;

    DateTime currentDateTime;
    currentDateTime.UniversalTime = ui.QuadPart;
    return currentDateTime;
}
```

Прочитайте Как получить текущую DateTime в C ++ UWP онлайн:
https://riptutorial.com/ru/uwp/topic/10131/как-получить-текущую-datetime-в-c-plusplus-uwp

глава 11: навигация

Вступление

В скором времени, когда приложение имеет несколько страниц / экранов, необходим способ навигации между ними. С приложениями UWP навигацию обрабатывает элемент управления [Frame] [1]. Он отображает экземпляры [Page] [2], поддерживает навигацию на новые страницы и сохраняет историю как для обратной и прямой навигации [1]: https://msdn.microsoft.com/en-us/library/windows/apps/ windows.ui.xaml.controls.frame.aspx [2]: https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.controls.page.aspx

Examples

Создать фрейм

Кадр создается как любые другие элементы управления:

```
<Frame x:Name="contentRoot"
       Navigated="OnNavigated"
       Navigating="OnNavigating" />
```

Затем навигационные / навигационные события могут быть перехвачены, чтобы отменить навигацию или показать / скрыть кнопку «Назад».

```
private void OnNavigating(object sender, NavigatingCancelEventArgs e)
{
    if(contentRoot.SourcePageType  == e.SourcePageType && m_currentPageParameter ==
e.Parameter)
   {
       // we are navigating again to the current page, we cancel the navigation
       e.Cancel    = true;
   }
}

private void OnNavigated(object sender, NavigationEventArgs e)
{
    // user has navigated to a newest page, we check if we can go back and show the back
button if needed.
    // we can also alter the backstack navigation history if needed
    SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility =
(contentRoot.CanGoBack ? AppViewBackButtonVisibility.Visible :
AppViewBackButtonVisibility.Collapsed);
}
```

Перейдите на новую страницу

Чтобы перейти на новую страницу, мы можем использовать метод Navigate () из фрейма.

```
contentRoot.Navigate(typeof(MyPage), parameter);
```

где `contentRoot` - это экземпляр Frame и `MyPage` - элемент управления, наследующий от страницы

В `MyPage` метод OnNavigatedTo () вызывается после завершения навигации (т.е. когда пользователь войдет на страницу), что позволяет нам запускать или завершать загрузку данных страницы. Метод OnNavigatedFrom () будет вызываться при выходе из страницы, что позволит нам освободить то, что должно быть выпущено.

```
public class MyPage : Page
{
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        // the page is now the current page of the application. We can finalized the loading
of the data to display
    }

    protected override void OnNavigatedFrom(NavigationEventArgs e)
    {
        // our page will be removed from the screen, we release what has to be released
    }
}
```

Подтверждение навигационного запроса с помощью OnNavigatingFrom

```
private bool navigateFlag = false;

protected async override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
    base.OnNavigatingFrom(e);

    if (!navigateFlag)
        {
            e.Cancel = true;

            var dialog = new MessageDialog("Navigate away?", Confir,);
            dialog.Commands.Add(new UICommand("Yes", null, 0));
            dialog.Commands.Add(new UICommand("No", null, 1);

            dialog.CancelCommandIndex = 1;
            dialog.DefaultCommandIndex = 0;

            var result = await dialog.ShowAsync();

            if (Convert.ToInt16(result.Id) != 1)
            {
                navigateFlag= true;
                this.Frame.Navigate(e.SourcePageType);
            }

        }

    }
```

Прочитайте навигация онлайн: https://riptutorial.com/ru/uwp/topic/8184/навигация

глава 12: Навигация по WebView

замечания

Все примеры, которые извлекают данные из удаленного URL-адреса, должны иметь возможность «Интернет (клиент)», указанную в Package.appxmanifest. Для примеров, которые управляют только локальными данными, нет необходимости.

Examples

Перейдите к Uri

Этот код просто перемещает WebView на некоторые Uri:

```
this.webView.Navigate(new Uri("http://stackoverflow.com/"));
```

или же

```
this.webView.Source = new Uri("http://stackoverflow.com/");
```

Навигация с помощью HttpRequestMessage

Установите пользовательский пользовательский агент и перейдите к Uri:

```
var userAgent = "my custom user agent";
var uri = new Uri("http://useragentstring.com/");
var requestMessage = new HttpRequestMessage(HttpMethod.Get, uri);
requestMessage.Headers.Add("User-Agent", userAgent);

this.webView.NavigateWithHttpRequestMessage(requestMessage);
```

Перейдите к строке

Показывать указанную строку html в WebView:

```
var htmlString =
    @"<!DOCTYPE html>
      <html>
          <head><title>HTML document</title></head>
          <body>
              <p>This is simple HTML content.</p>
          </body>
      </html>";

this.webView.NavigateToString(htmlString);
```

Открыть HTML-файл из пакета приложений

Вы можете легко открыть файл из своего пакета приложений, но схема Uri должна быть « ms-appx-web» вместо «ms-appx»:

```
var uri = new Uri("ms-appx-web:///Assets/Html/html-sample.html");
this.webView.Navigate(uri);
```

Открыть HTML-файл из локальной папки приложения или папки temp

Чтобы открыть файл из локальной папки или папки temp, целевой файл **не должен** находиться в корне каталога. Из соображений безопасности, чтобы предотвратить показ другого контента WebView, файл, предназначенный для отображения, должен быть расположен во вложенной папке:

```
var uri = new Uri("ms-appdata:///local/html/html-sample.html");
this.webView.Navigate(uri);
```

NavigateToLocalStreamUri

Если NavigateToString не может обрабатывать какой-либо контент, используйте метод NavigateToLocalStreamUri. Это заставит каждый локализованный URI внутри HTML-страницы вызвать специальный класс распознавателя, который может обеспечить правильное содержимое «на лету».

Assets / Html / html-sample.html файл:

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML document</title>
    </head>
    <body>
        <p>This is simple HTML content.</p>
        <img src="cat.jpg"/>
    </body>
</html>
```

Код:

```
protected override void OnNavigatedTo(NavigationEventArgs args)
{
    // The Uri resolver takes is in the form of "ms-local-stream://appname_KEY/folder/file"
    // For simplicity, there is method BuildLocalStreamUri which returns correct Uri.
    var uri = this.webView.BuildLocalStreamUri("SomeTag", "/html-sample.html");
    var resolver = new StreamUriResolver();
    this.webView.NavigateToLocalStreamUri(uri, resolver);

    base.OnNavigatedTo(args);
}
```

```
public sealed class StreamUriResolver : IUriToStreamResolver
{
    public IAsyncOperation<IInputStream> UriToStreamAsync(Uri uri)
    {
        if (uri == null)
        {
            throw new ArgumentNullException(nameof(uri));
        }

        var path = uri.AbsolutePath;

        return GetContent(path).AsAsyncOperation();
    }


    private async Task<IInputStream> GetContent(string uriPath)
    {
        Uri localUri;

        if (Path.GetExtension(uriPath).Equals(".html"))
        {
            localUri = new Uri("ms-appx:///Assets/Html" + uriPath);
        }
        else
        {
            localUri = new Uri("ms-appdata:///local/content" + uriPath);
        }

        var file = await StorageFile.GetFileFromApplicationUriAsync(localUri);
        var stream = await file.OpenAsync(FileAccessMode.Read);

        return stream.GetInputStreamAt(0);
    }
}
```

Этот код будет принимать HTML-страницу из пакета приложений и вставлять содержимое из локальной папки в нее. Если у вас есть образ «cat.jpg» в папке / local / content, он покажет HTML-страницу с изображением кошки.

Прочитайте Навигация по WebView онлайн: https://riptutorial.com/ru/uwp/topic/6321/навигация-по-webview

глава 13: Настройки и данные приложения

Examples

Сохранять и получать настройки

Приложения UWP могут легко хранить простые настройки в хранилище ключей / значений локально или даже в облаке, поэтому ваше приложение или игра могут совместно использовать настройки между различными устройствами пользователя.

Для настроек можно использовать следующие типы данных:

- UInt8, Int16, UInt16, Int32, UInt32, Int64, UInt64, Single, Double
- логический
- Char16, String
- DateTime, TimeSpan
- GUID, Point, Size, Rect

Начните с получения локального и / или перемещаемого контейнера данных.

```
Windows.Storage.ApplicationDataContainer localSettings =
Windows.Storage.ApplicationData.Current.LocalSettings;
Windows.Storage.ApplicationDataContainer roamingSettings =
Windows.Storage.ApplicationData.Current.RoamingSettings;
```

Чтобы создать или записать параметр, используйте свойство ApplicationDataContainer.Values для доступа к настройкам в контейнере данных. Например, создайте локальный параметр с именем `FontSize` с `FontSize int 10` и параметром роуминга `Username` со `string` значением `Bob` .

```
localSettings.Values["FontSize"] = 10;
roamingSettings.Values["Username"] = "Bob";
```

Чтобы получить этот параметр, используйте тот же объект ApplicationDataContainer.Values , который использовался для создания этого параметра.

```
int fontSize = localSettings["FontSize"];
string username = roamingSettings["Username"];
```

Хорошая практика - проверить, существует ли настройка до ее получения.

```
if (localSettings.Values.ContainsKey("FontSize"))
    int fontSize = localSettings["FontSize"];

if (roamingSettings.Values.ContainsKey("Username"))
    string username = roamingSettings["Username"];
```

Настройки роуминга имеют квоту размера. Используйте свойство RoamingStorageQuota, чтобы получить его.

Подробнее о настройках, их ограничениях и примерах кода можно узнать в MSDN .

Сохранение данных в кеш приложения

ApplicationData.Current.LocalFolder api позволяет нам получить доступ к кешу приложения:

```
var file = await ApplicationData.Current.LocalFolder.CreateFileAsync("myFile.dat",
CreationCollisionOption.ReplaceExisting);
```

Класс FileIO содержит набор методов утилиты для простого добавления данных в файл:

```
await FileIO.WriteBytesAsync(file, array);
await FileIO.AppendTextAsync(file, "text");
await FileIO.WriteBufferAsync(file, iBuffer);
```

Прочитайте Настройки и данные приложения онлайн:
https://riptutorial.com/ru/uwp/topic/5944/настройки-и-данные-приложения

глава 14: Отборочные имена файлов

замечания

Квалификаторы используются в этом общем формате:

Файлы : `filename.qualifier-value.ext`
~ несколько квалификаторов: `filename.qualifier1-value1_qualifier2-value2_....ext`

Квалифицированные папки: `qualifier-value`
~ несколько квалификаторов: `qualifier1-value1_qualifier2-value2_...`

Квалификаторы перечислены ниже, они используются в формате, описанном выше

спецификатор	использование	Ценности
Язык / Язык	Указывает язык, регион или и то, и другое.	`xx-xx` или `xx` в BCP-47
Масштаб	Определяет коэффициент масштабирования устройства.	*Обычно 100/125/150/200/400*
DeviceFamily	Указывает тип устройства.	Мобильный / Team / Desktop / IoT
контрастировать	Определяет тип темы контраста.	Стандарт / Высокий / Черный / Белый
HomeRegion	Указывает домашний регион пользователя.	Любой ISO3166-1 alpha2 или цифровой код
TargetSize	Дает наименьшее изображение больше, чем нужно.	Любое положительное целое число.
LayoutDir	Определяет направление макета.	RTL / LTR / TTBRTL / TTBLTR
конфиг	Квалифицируется для `MS_CONFIGURATION_ATTRIBUTE_VALUE` .	Значение конфигурации среды.
DXFL *	Определяет уровень функции DirectX.	DX9 / DX10 / DX11

* Также используется как **DXFeatureLevel** .

Некоторые примечания должны иметь в виду:

- `HomeRegion` не будет принимать группы или союзы.
- `TargetSize` и `Scale` не могут использоваться вместе.

Examples

Использование разных видов для типов устройств

Вы можете определить всю папку папок для определенного типа устройства, ее файлы переопределяют те, что находятся за ее пределами на этом устройстве:

```
/ DeviceFamily-Mobile
    PageOfEden.xaml
    MainPage.xaml
MainPage.xaml
MainPage.xaml.cs
PageOfEden.xaml
PageOfEden.xaml.cs
```

Файлы внутри отборочной папки не нуждаются в квалификаторах.

Отборочные критерии масштабирования по умолчанию

Если вы просматриваете папку « *Активы* » вашего приложения, вы заметите, что все ресурсы квалифицируются по их масштабам (поскольку вам необходимо поместить отдельные файлы для каждого масштабирования в манифесте пакета).

```
SplashScreen.scale-100.png
SplashScreen.scale-125.png
SplashScreen.scale-150.png
SplashScreen.scale-200.png
```

Использование определителя TargetSize

Предположим, у нас есть элемент *Image,* используя квадратное изображение с именем `Picture.png` .
Мы можем использовать разные файлы для каждого набора измерений для элемента.

```
Picture.TargetSize-16.png
Picture.TargetSize-32.png
Picture.TargetSize-128.png
```

Теперь, если мы установим `Height` или `Width` нашего *изображения* на 16 `Picture.TargetSize-16.png` в качестве источника будет использоваться `Picture.TargetSize-16.png` . Теперь, если

мы установим размеры 20px, изображение не будет соответствовать точным размерам, поэтому оно будет использовать `Picture.TargetSize-32.png` , так как это ближайшее изображение больше, чем наши потребности. Размеры выше 128 будут использовать `Picture.TargetSize-128.png` .

Прочитайте Отборочные имена файлов онлайн: https://riptutorial.com/ru/uwp/topic/6733/отборочные-имена-файлов

глава 15: Преобразование размера изображения и файла изображения обрезки в приложении Windows Universal

Examples

Обрезать и изменять размер изображения с помощью растрового инструмента

```
 public class BitmapTools
    {
        /// <summary>
        /// Gets the cropped bitmap asynchronously.
        /// </summary>
        /// <param name="originalImage">The original image.</param>
        /// <param name="startPoint">The start point.</param>
        /// <param name="cropSize">Size of the corp.</param>
        /// <param name="scale">The scale.</param>
        /// <returns>The cropped image.</returns>
        public static async Task<WriteableBitmap> GetCroppedBitmapAsync(IRandomAccessStream
originalImage,
            Point startPoint, Size cropSize, double scale)
        {
            if (double.IsNaN(scale) || double.IsInfinity(scale))
            {
                scale = 1;
            }

            // Convert start point and size to integer.
            var startPointX = (uint)Math.Floor(startPoint.X * scale);
            var startPointY = (uint)Math.Floor(startPoint.Y * scale);
            var height = (uint)Math.Floor(cropSize.Height * scale);
            var width = (uint)Math.Floor(cropSize.Width * scale);

            // Create a decoder from the stream. With the decoder, we can get
            // the properties of the image.
            var decoder = await BitmapDecoder.CreateAsync(originalImage);

            // The scaledSize of original image.
            var scaledWidth = (uint)Math.Floor(decoder.PixelWidth * scale);
            var scaledHeight = (uint)Math.Floor(decoder.PixelHeight * scale);

            // Refine the start point and the size.
            if (startPointX + width > scaledWidth)
            {
                startPointX = scaledWidth - width;
            }

            if (startPointY + height > scaledHeight)
            {
                startPointY = scaledHeight - height;
            }
```

```
            // Get the cropped pixels.
            var pixels = await GetPixelData(decoder, startPointX, startPointY, width, height,
                scaledWidth, scaledHeight);

            // Stream the bytes into a WriteableBitmap
            var cropBmp = new WriteableBitmap((int)width, (int)height);
            var pixStream = cropBmp.PixelBuffer.AsStream();
            pixStream.Write(pixels, 0, (int)(width * height * 4));

            return cropBmp;
        }

        /// <summary>
        /// Gets the pixel data.
        /// </summary>
        /// <remarks>
        /// If you want to get the pixel data of a scaled image, set the scaledWidth and
scaledHeight
        /// of the scaled image.
        /// </remarks>
        /// <param name="decoder">The bitmap decoder.</param>
        /// <param name="startPointX">The X coordinate of the start point.</param>
        /// <param name="startPointY">The Y coordinate of the start point.</param>
        /// <param name="width">The width of the source rect.</param>
        /// <param name="height">The height of the source rect.</param>
        /// <param name="scaledWidth">The desired width.</param>
        /// <param name="scaledHeight">The desired height.</param>
        /// <returns>The image data.</returns>
        private static async Task<byte[]> GetPixelData(BitmapDecoder decoder, uint
startPointX, uint startPointY,
            uint width, uint height, uint scaledWidth, uint scaledHeight)
        {
            var transform = new BitmapTransform();
            var bounds = new BitmapBounds();
            bounds.X = startPointX;
            bounds.Y = startPointY;
            bounds.Height = height;
            bounds.Width = width;
            transform.Bounds = bounds;

            transform.ScaledWidth = scaledWidth;
            transform.ScaledHeight = scaledHeight;

            // Get the cropped pixels within the bounds of transform.
            var pix = await decoder.GetPixelDataAsync(
                BitmapPixelFormat.Bgra8,
                BitmapAlphaMode.Straight,
                transform,
                ExifOrientationMode.IgnoreExifOrientation,
                ColorManagementMode.ColorManageToSRgb);
            var pixels = pix.DetachPixelData();
            return pixels;
        }

        /// <summary>
        /// Resizes the specified stream.
        /// </summary>
        /// <param name="sourceStream">The source stream to resize.</param>
        /// <param name="newWidth">The width of the resized image.</param>
        /// <param name="newHeight">The height of the resized image.</param>
```

```
        /// <returns>The resized image stream.</returns>
        public static async Task<InMemoryRandomAccessStream> Resize(IRandomAccessStream
sourceStream, uint requestedMinSide)
        {
            var decoder = await BitmapDecoder.CreateAsync(sourceStream);
            uint originalPixelWidth = decoder.OrientedPixelWidth;
            uint originalPixelHeight = decoder.OrientedPixelHeight;

            double widthRatio = (double)requestedMinSide / originalPixelWidth;
            double heightRatio = (double)requestedMinSide / originalPixelHeight;
            uint aspectHeight = (uint)requestedMinSide;
            uint aspectWidth = (uint)requestedMinSide;
            var scaledSize = (uint)requestedMinSide;
            if (originalPixelWidth < originalPixelHeight)
            {
                aspectWidth = (uint)(heightRatio * originalPixelWidth);
            }
            else
            {
                aspectHeight = (uint)(widthRatio * originalPixelHeight);
            }

            var destinationStream = new InMemoryRandomAccessStream();


            var transform = new BitmapTransform { ScaledWidth = aspectWidth, ScaledHeight =
aspectHeight };

            var pixelData = await decoder.GetPixelDataAsync(
                BitmapPixelFormat.Bgra8,
                BitmapAlphaMode.Straight,
                transform,
                ExifOrientationMode.RespectExifOrientation,
                ColorManagementMode.DoNotColorManage);

            var encoder =
                await BitmapEncoder.CreateAsync(BitmapEncoder.JpegEncoderId,
destinationStream);

            if(decoder.OrientedPixelHeight!=decoder.PixelHeight &&
decoder.OrientedPixelWidth!=decoder.PixelWidth)
            encoder.BitmapTransform.Rotation = BitmapRotation.Clockwise270Degrees;

            encoder.SetPixelData(BitmapPixelFormat.Bgra8, BitmapAlphaMode.Premultiplied,
aspectWidth, aspectHeight, 96, 96,
                pixelData.DetachPixelData());
            await encoder.FlushAsync();


            return destinationStream;
        }

        /// <summary>
        /// Rotates the given stream.
        /// </summary>
        /// <param name="randomAccessStream">The random access stream.</param>
        /// <param name="rotation">The rotation.</param>
        /// <returns>The stream.</returns>
        public static async Task<InMemoryRandomAccessStream> Rotate(IRandomAccessStream
randomAccessStream,
            BitmapRotation rotation)
```

```
        {
            var decoder = await BitmapDecoder.CreateAsync(randomAccessStream);

            var rotatedStream = new InMemoryRandomAccessStream();

            var encoder = await BitmapEncoder.CreateForTranscodingAsync(rotatedStream,
decoder);

            encoder.BitmapTransform.Rotation = rotation;
            encoder.BitmapTransform.InterpolationMode = BitmapInterpolationMode.Fant;

            await encoder.FlushAsync();

            return rotatedStream;
        }


        /// <summary>
        /// Resizes and crops source file image so that resized image width/height are not
larger than <param name="requestedMinSide"></param>
        /// </summary>
        /// <param name="sourceFile">Source StorageFile</param>
        /// <param name="requestedMinSide">Width/Height of the output image</param>
        /// <param name="resizedImageFile">Target StorageFile</param>
        /// <returns></returns>
        public static async Task<IStorageFile> CreateThumbnaiImage(StorageFile sourceFile, int
requestedMinSide, StorageFile resizedImageFile)
        {
            var imageStream = await sourceFile.OpenReadAsync();
            var decoder = await BitmapDecoder.CreateAsync(imageStream);
            var originalPixelWidth = decoder.PixelWidth;
            var originalPixelHeight = decoder.PixelHeight;
            using (imageStream)
            {
                //do resize only if needed
                if (originalPixelHeight > requestedMinSide && originalPixelWidth >
requestedMinSide)
                {
                    using (var resizedStream = await
resizedImageFile.OpenAsync(FileAccessMode.ReadWrite))
                    {
                        //create encoder based on decoder of the source file
                        var encoder = await
BitmapEncoder.CreateForTranscodingAsync(resizedStream, decoder);
                        double widthRatio = (double)requestedMinSide / originalPixelWidth;
                        double heightRatio = (double)requestedMinSide / originalPixelHeight;
                        uint aspectHeight = (uint)requestedMinSide;
                        uint aspectWidth = (uint)requestedMinSide;
                        uint cropX = 0, cropY = 0;
                        var scaledSize = (uint)requestedMinSide;
                        if (originalPixelWidth > originalPixelHeight)
                        {
                            aspectWidth = (uint)(heightRatio * originalPixelWidth);
                            cropX = (aspectWidth - aspectHeight) / 2;
                        }
                        else
                        {
                            aspectHeight = (uint)(widthRatio * originalPixelHeight);
                            cropY = (aspectHeight - aspectWidth) / 2;
                        }
                        //you can adjust interpolation and other options here, so far linear
```

```
is fine for thumbnails
                        encoder.BitmapTransform.InterpolationMode =
BitmapInterpolationMode.Linear;
                        encoder.BitmapTransform.ScaledHeight = aspectHeight;
                        encoder.BitmapTransform.ScaledWidth = aspectWidth;
                        encoder.BitmapTransform.Bounds = new BitmapBounds()
                        {
                            Width = scaledSize,
                            Height = scaledSize,
                            X = cropX,
                            Y = cropY,
                        };
                        await encoder.FlushAsync();
                    }
                }
                else
                {
                    //otherwise just use source file as thumbnail
                    await sourceFile.CopyAndReplaceAsync(resizedImageFile);
                }
            }
            return resizedImageFile;
        }

    }
```

Прочитайте Преобразование размера изображения и файла изображения обрезки в приложении Windows Universal онлайн: https://riptutorial.com/ru/uwp/topic/9529/преобразование-размера-изображения-и-файла-изображения-обрезки-в-приложении-windows-universal

глава 16: Работа с файловой системой

Examples

Как обмениваться данными на нескольких устройствах в приложении Win10 UWP

Чтобы сделать приложение более сплоченным, нам часто нужно поддерживать индивидуальные настройки и предпочтения пользователя на нескольких устройствах, которые вошли в систему с одной учетной записью Microsoft. В этом примере мы используем данные роуминга для хранения и загрузки настроек пользовательского интерфейса, игрового процесса и информации о пользователе. Но данные роуминга имеют собственный предел: мы не можем хранить большой файл в папке роуминга. Система приостанавливает репликацию данных для всех приложений в пакете до облака до тех пор, пока текущий размер больше не будет превышать максимальный размер. Поэтому в этом примере мы не сохранили изображение пользователя в папке роуминга. Вместо этого он сохраняется в локальной папке.

```
private async void LoadRoamingData()
{
    //Get background color
    object color = roamingSettings.Values["BackgroundColor"];
    if (color != null)
    {
        if (ViewModel.ColorList.Keys.Contains(color.ToString()))
        {
            Color backgroundColor = ViewModel.ColorList[color.ToString()];
            ViewModel.BackgroundColor = new SolidColorBrush(backgroundColor);
            comboBackgroundColor.SelectedValue = color.ToString();
        }
    }
    //Get game process stored in the roaming file
    try
    {
        StorageFile processFile = await roamingFolder.GetFileAsync(processFileName);
        string process = await FileIO.ReadTextAsync(processFile);
        int gameProcess;
        if (process != null && int.TryParse(process.ToString(), out gameProcess) &&
gameProcess > 0)
        {
            ViewModel.GameProcess = gameProcess;
        }
    }
    catch { }

    //Get user name
    object userName = roamingSettings.Values["UserName"];
    if (userName != null && !string.IsNullOrWhiteSpace(userName.ToString()))
    {
        ViewModel.UserName = userName.ToString();
    }
```

```
}
```

Для получения дополнительной информации см. Https://code.msdn.microsoft.com/How-to-share-data-across-d492cc0b .

Прочитайте Работа с файловой системой онлайн: https://riptutorial.com/ru/uwp/topic/6480/работа-с-файловой-системой

глава 17: Ресурсы в UWP (StaticResource / ThemeResource) и ResourceDictionary

Вступление

В новых приложениях Windows 10 существует много способов ссылки на ресурс внутри кода XAML или кода. Прежде всего, вы должны объявить ресурсы в каком-то доступном месте. Легкий способ - объявить `ResourceDictionary` в контексте, скажем, на текущей странице.

Examples

1. Словарь ресурсов

Фрагмент из MainPage.xaml

```
<Page
    x:Class="MyNewApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MyNewApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Page.Resources>
        <!-- Creates a resource dictionary in the page context -->
        <ResourceDictionary>
            <!-- This is a solid color brush resource
                 NOTE: Each resource inside a resource dictionary must have a key -->
            <SolidColorBrush x:Key="ColorRed">Red</SolidColorBrush>
        </ResourceDictionary>
    </Page.Resources>

    <!-- Using ThemeResource in here to access a resource already defined -->
    <Grid Background="{ThemeResource ColorRed}">

    </Grid>
</Page>
```

2. Глобальные ресурсы

Ресурсные словари доступны только внутри контекста, который был объявлен, поэтому, если мы планируем ссылаться на ресурсы, объявленные в одном контексте страницы с другой страницы, они не будут найдены. Поэтому, если нам нужны глобальные ресурсы,

которые будут определены как те, которые поставляются с каркасом, мы делаем это в App.xaml

Сниппет из App.xaml

```
<Application
    x:Class="MyNewApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    RequestedTheme="Dark">

    <Application.Resources>
        <ResourceDictionary>
            <SolidColorBrush x:Key="ColorRed">Red</SolidColorBrush>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

Таким образом, мы можем получить доступ `ColorRed` цветному ресурсу ColorRed из любого приложения в нашем приложении. Но подождите, мы не хотим заражать этот маленький файл всеми ресурсами нашего приложения! Итак, мы делаем `MergedDictionaries`

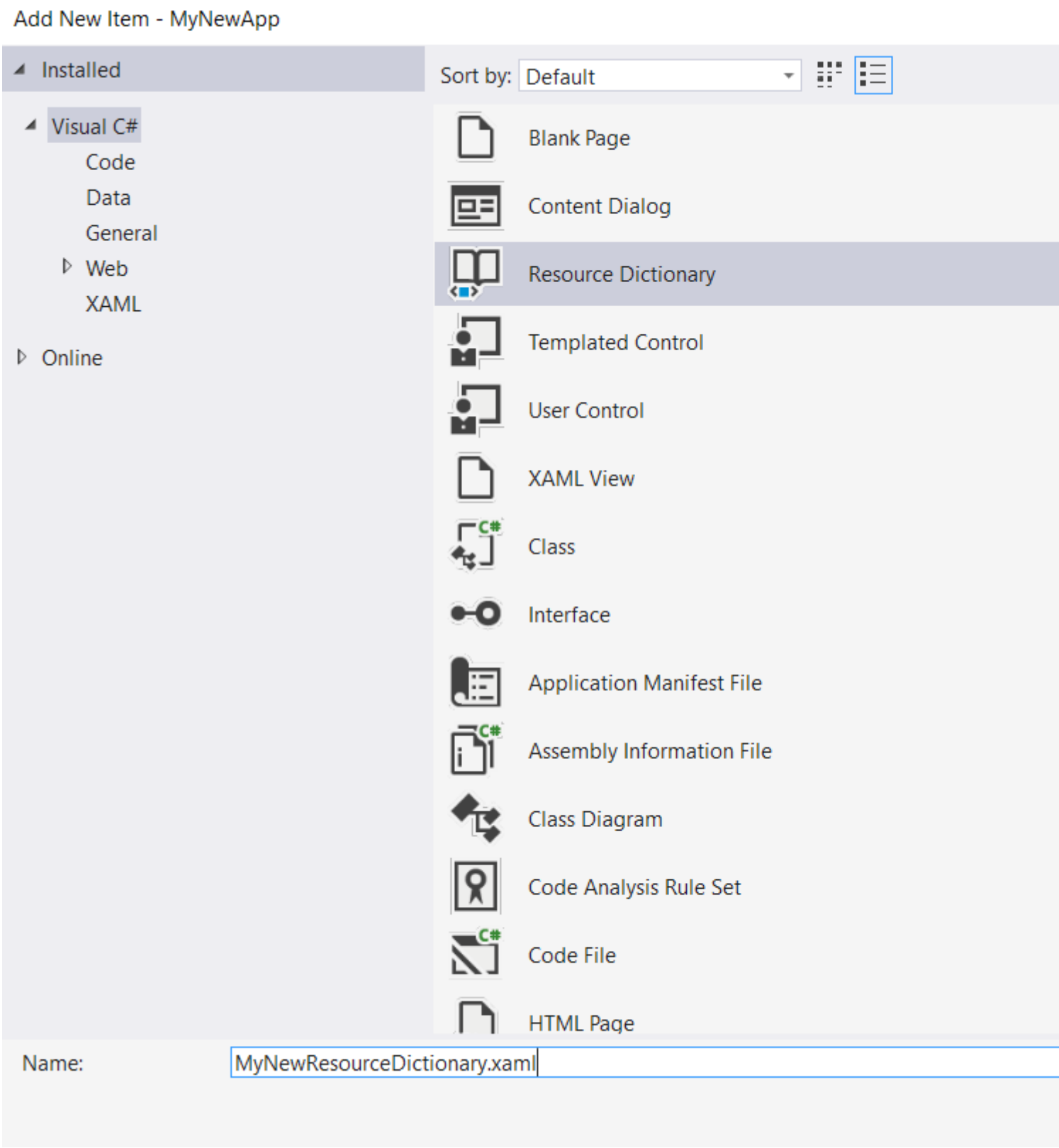
3. Объединенные словари

Практически обычно вещи немного сложнее, и для поддержки масштабируемости мы должны разделить вещи друг от друга. Таким образом, мы можем определить различные файлы, содержащие различные словари ресурсов, то есть ресурсы для тем оформления элементов управления пользовательского интерфейса, ресурсы для текстов и т. Д., Затем мы объединим их все вместе в файле App.xaml.

Сниппет из App.xaml

```
<Application
    x:Class="MyNewApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    RequestedTheme="Dark">

    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/Assets/Themes/GeneralStyles.xaml"/>
                <ResourceDictionary Source="/Assets/Themes/TemplatedControls.xaml"/>
                <ResourceDictionary Source="/Assets/Strings/Texts.xaml"/>
                <ResourceDictionary Source="/Assets/Strings/ErrorTexts.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

Вы можете создать новый файл словаря, щелкнув правой кнопкой мыши папку «Активы» [Добавить -> Новый элемент]



4. Доступ к ресурсам

Теперь нам нужно получить доступ к объявленным ресурсам, чтобы сделать это из кода

XAML, который мы используем `{ThemeResource ResourceKey}` или `{StaticResource ResourceKey}`

```
to be continued later.
```

Прочитайте Ресурсы в UWP (StaticResource / ThemeResource) и ResourceDictionary онлайн: https://riptutorial.com/ru/uwp/topic/10511/ресурсы-в-uwp--staticresource---themeresource--и-resourcedictionary

глава 18: Связывание vs x: Bind

Синтаксис

- <object property = "{x: Bind}" ... />
- <object property = "{x: Bind propertyPath}" ... />
- <object property = "{x: Bind Path = propertyPath}" ... />
- <object property = "{x: Bind [bindingProperties]}" ... />
- <object property = "{x: Bind propertyPath, [bindingProperties]}" ... />
- <object property = "{x: Bind Path = propertyPath, [bindingProperties]}" ... />

замечания

Расширение разметки `{x:Bind}` - новое для Windows 10 - является альтернативой `{Binding}` .

`{x:Bind}` не хватает некоторых функций `{Binding}` , но работает меньше времени и меньше памяти, чем `{Binding}` и поддерживает лучшую отладку.

При времени загрузки XAML `{x:Bind}` преобразуется в то, что вы можете рассматривать как объект привязки, и этот объект получает значение из свойства в источнике данных. Объект привязки может быть дополнительно настроен для наблюдения за изменением значения свойства источника данных и обновления на основе этих изменений. Он также может быть настроен на то, чтобы вносить изменения в свое значение обратно в исходное свойство. Объекты привязки, созданные `{x:Bind}` и `{Binding}` , в основном функционально эквивалентны. Но `{x:Bind}` выполняет специальный код, который он генерирует во время компиляции, а `{Binding}` использует проверку объектов общего назначения. Следовательно, привязки `{x:Bind}` (часто называемые скомпилированными связями) обладают большой производительностью, обеспечивают проверку ваших выражений привязки во время компиляции и поддерживают отладку, позволяя вам устанавливать контрольные точки в файлах кода, которые генерируются как частичные класс для вашей страницы. Эти файлы можно найти в вашей папке obj с именами вроде (для C #) .g.cs.

Для получения дополнительной информации см. Документацию MSDN по адресу x: Bind

Examples

Оценка {x: Bind} из функций

Эта способность была добавлена к расширению разметки Bind после v1607 (Redstone 1).

Вы можете указать путь к функции, а также пути arg и константные args. Предположим, что у нас есть функция, определенная в нашем обратном коде:

```
public string Translate(string text, string from, string to)
```

Теперь мы можем использовать bind для оценки функции в нужном элементе:

```
<TextBlock Name="SomeText" Text="How are you?" />
<TextBlock Name="{x:Bind Translate(SomeText.Text, 'en', 'es')}" />
```

Функции и пути arg также могут содержать точки и заливки.

Прочитайте Связывание vs x: Bind онлайн: https://riptutorial.com/ru/uwp/topic/4951/связывание-vs-x--bind

глава 19: Связывание vs x: Bind

замечания

Обратитесь к официальной документации, связанной с данными, от Microsoft.

Examples

Режимы привязки и значения по умолчанию

Существует три способа привязки XAML для `Binding` и `x:Bind` :

- **OneTime** : обновление происходит только один раз при инициализации представления во время вызова `InitializeComponent()` . *(ViewModel [отправляет данные при инициализации] -> Просмотр)*
- **OneWay** : View обновляется при изменении ViewModel. Но не в обратном направлении. *(ViewModel -> Просмотр)*
- **TwoWay** : View обновляется при изменении ViewModel и наоборот. *(ViewModel <-> Вид)*

По умолчанию режим `Binding` является `OneWay` и что из `x:Bind` является `OneTime` .

Выберите такие режимы, как:

```
<TextBlock Text="{Binding SomeText, Mode=TwoWay}" /> <!-- Binding -->
<TextBlock Text="{x:Bind SomeText, Mode=OneWay}" /> <!-- x:Bind -->
```

Когда использовать x: Bind

- При вызове методов непосредственно из представления.
- Если производительность имеет значение очень плохо (научный материал космического корабля)
- Когда вы хотите получить ошибки времени компиляции

Когда использовать привязку

- Используйте его, если хотите быть гибкими в отношении типа источника данных. Он не будет привязан к фактическому свойству, а к его названию.
- Если вы хотите привязать к DataContext

Прочитайте Связывание vs x: Bind онлайн: https://riptutorial.com/ru/uwp/topic/6412/связывание-vs-x--bind

глава 20: Семейство устройств

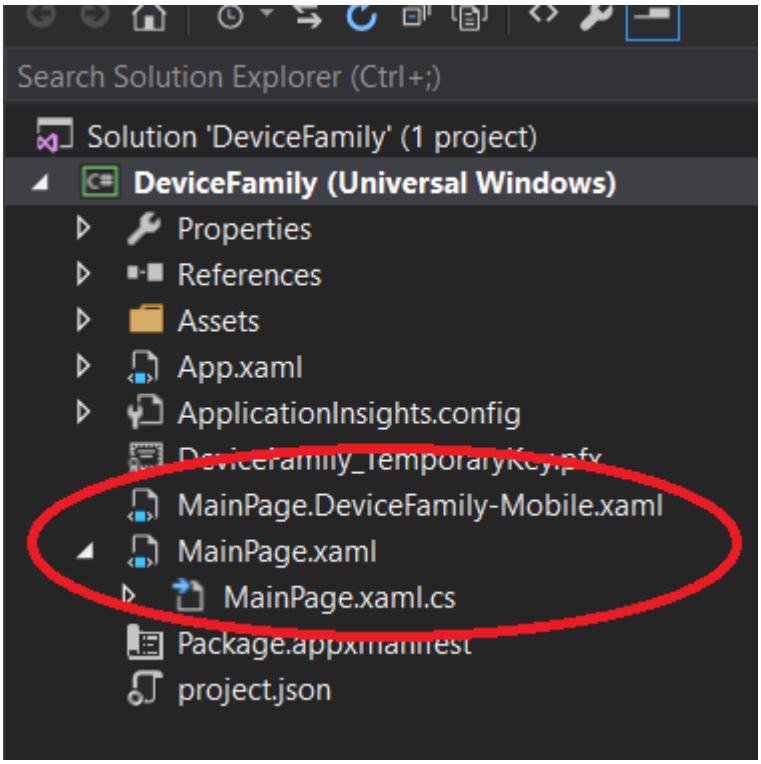
Examples

DeviceFamily специальный код

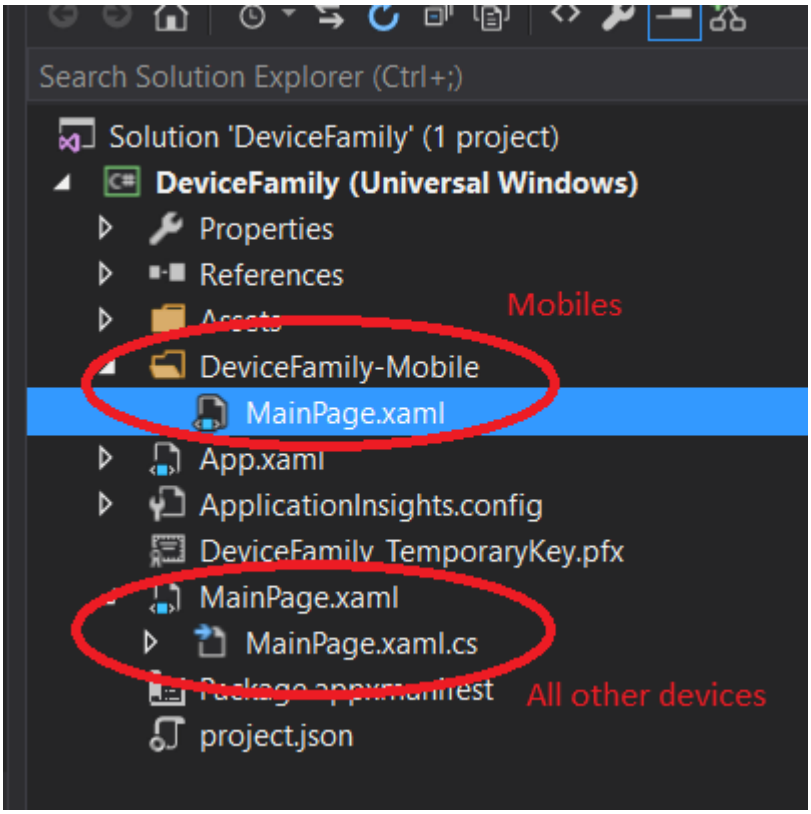
В общем, UWP используется для создания одного приложения, которое выполняется в Windows 10 на разных устройствах. Тем не менее, также возможно сделать код, адаптированный к конкретным устройствам. Вы можете достичь этого несколькими способами.

Различные макеты XAML

Если вы хотите использовать определенный макет для определенного «семейства устройств», вы можете сделать это, создав новый элемент страницы XAML с тем же именем, что и файл XAML по умолчанию, с суффиксом, чтобы указать семейство устройств, на которое вы нацеливаете. Затем у вас будет *MainPage.xaml* для всех устройств и *MainPage.DeviceFamily- [определенного семейства] .xaml* только для одного определенного семейства, которое перезапишет макет по умолчанию, см. Ниже:



Если вы хотите сделать это для большого количества файлов, вы можете создать папку с именем *DeviceFamily- [конкретное семейство]* и поместить в нее все страницы XAML, но теперь точно с тем же именем, что и файл XAML по умолчанию (см. Ниже). В обоих примерах все страницы будут иметь один и тот же файл с кодом, поэтому функциональность идентична, но макет соответствует конкретным размерам экрана.



Код для конкретной семьи

Если вы хотите запустить часть кода или вашего ViewModel только для определенного семейства устройств, вы можете использовать свойство `DeviceFamily` из класса `AnalyticsVersionInfo` .

```
AnalyticsVersionInfo avi = AnalyticsInfo.VersionInfo;
var deviceFamily = avi.DeviceFamily;

if(deviceFamily == "Windows.Mobile")
{
   Console.WriteLine("You're on mobile device right now.");
}
else if(deviceFamily == "Windows.Desktop")
{
   Console.WriteLine("You're on desktop");
}
else if(deviceFamily == "Windows.IoT")
{
   Console.WriteLine("You're on IoT");
}
//....
```

Получить текущее семейство устройств

Вот простой переносной способ получить текущее семейство устройств:

```
/// <summary>
/// All the device families
/// </summary>
public enum DeviceFamily
```

```
{
    Desktop,
    Mobile,
    Iot,
    Xbox,
}

/// <summary>
/// The helper to get the current device family
/// </summary>
public static class DeviceFamilyHelper
{
    /// <summary>
    /// Return the family of the current device
    /// </summary>
    /// <returns>the family of the current device</returns>
    public static DeviceFamily GetDeviceFamily()
    {

switch(ResourceContext.GetForCurrentView().QualifierValues["DeviceFamily"].ToLowerInvariant())
        {
            case "mobile":  return DeviceFamily.Mobile;
            case "xbox":    return DeviceFamily.Xbox;
            case "iot":     return DeviceFamily.Iot;
            default:        return DeviceFamily.Desktop;
        }
    }
}
```

Обнаруживать, поддерживается ли контракт API

В зависимости от версии устройства / выпуска системы некоторые API могут быть недоступны. Вы можете проверить, какой контракт поддерживается с помощью ApiInformation.IsApiContractPresent ()

Например, это вернет true на телефонных устройствах и ложно на других

```
ApiInformation.IsApiContractPresent(typeof(CallsPhoneContract).FullName, 1)
```

Контракт, в котором находится API, доступен внизу страницы API в MSDN или глобальном списке, доступном на странице контракта API .

Прочитайте Семейство устройств онлайн: https://riptutorial.com/ru/uwp/topic/3994/семейство-устройств

глава 21: Тематические ресурсы

Синтаксис

- C #: Application.Current.Resources ["yourColorKey"]
- Xaml: {ThemeResource yourColorKey}

параметры

параметр	Цель
`yourColorKey`	Ключ, который вы даете, чтобы вернуть объект `Color` . Он отличается между C # и Xaml

замечания

UWP позволяет вам полностью контролировать преимущества Windows 10. Некоторые из этих преимуществ являются графическими, такими как цвет Accent или Dark / Light.

Чтобы подготовить приложение к совместимости с этими функциями, в UWP была реализована куча готовых цветов для изменения цвета Accent ОС, на которых работает программа, или с выбором темы для пользователя.

Это можно сделать двумя способами:

- Diretly в Xaml, используя атрибут `Color = {ThemeResource x}` (или любой другой атрибут, который принимает значение `Brush` как значение, например BorderBrush, Background и т. Д.),
- В C # Code Behind, путем поиска цвета в каталоге ресурсов текущего приложения. Это дает объект `Color` , поэтому, если вы хотите поместить его в свойство `Color` объекта, на который вы ссылаетесь, из вашего Xaml, вам нужно сделать новую кисть следующим образом:

```
new SolidColorBrush(Application.Current.Resources["yourColorKey"])
```

Для справки цветных клавиш в c #, пожалуйста, обратитесь к:

https://msdn.microsoft.com/windows/uwp/controls-and-patterns/xaml-theme-resources

Examples

Доступ к ресурсам темы в Xaml

Фрагмент из MyExampleFile.xaml

```
<TextBlock Foreground="{ThemeResource SystemControlBackgroundAccentBrush}"
         Text="This is a colored textbox that use the Accent color of your Windows 10"/>

<TextBlock Foreground="{ThemeResource SystemControlBackgroundBaseHighBrush}"
         Text="This is a colored textbox that use a color that is readable in both Light and
Dark theme"/>
```

Доступ к ресурсам темы в C

Фрагмент из MyExampleFile.xaml

```
<TextBlock x:Name="MyTextBlock"
         Text="This is a TextBlock colored from the code behind"/>
```

Фрагмент из MyExampleFile.xaml.cs

```
    // We use the application's Resource dictionary to get the current Accent of your Windows
10
    MyTextBlock.Color = new
SolidColorBrush(Application.Current.Resources["SystemAccentColor"]);
```

Прочитайте Тематические ресурсы онлайн: https://riptutorial.com/ru/uwp/topic/7527/тематические-ресурсы

глава 22: Тестирование модулей для UWP

Вступление

Я хотел бы показать вам, как создавать модульные тесты для универсального приложения Windows 10. Для тестирования приложений UWP мы будем использовать xUnit.net Framework, о которых вы можете прочитать больше из ссылки, представленной в разделе замечаний.

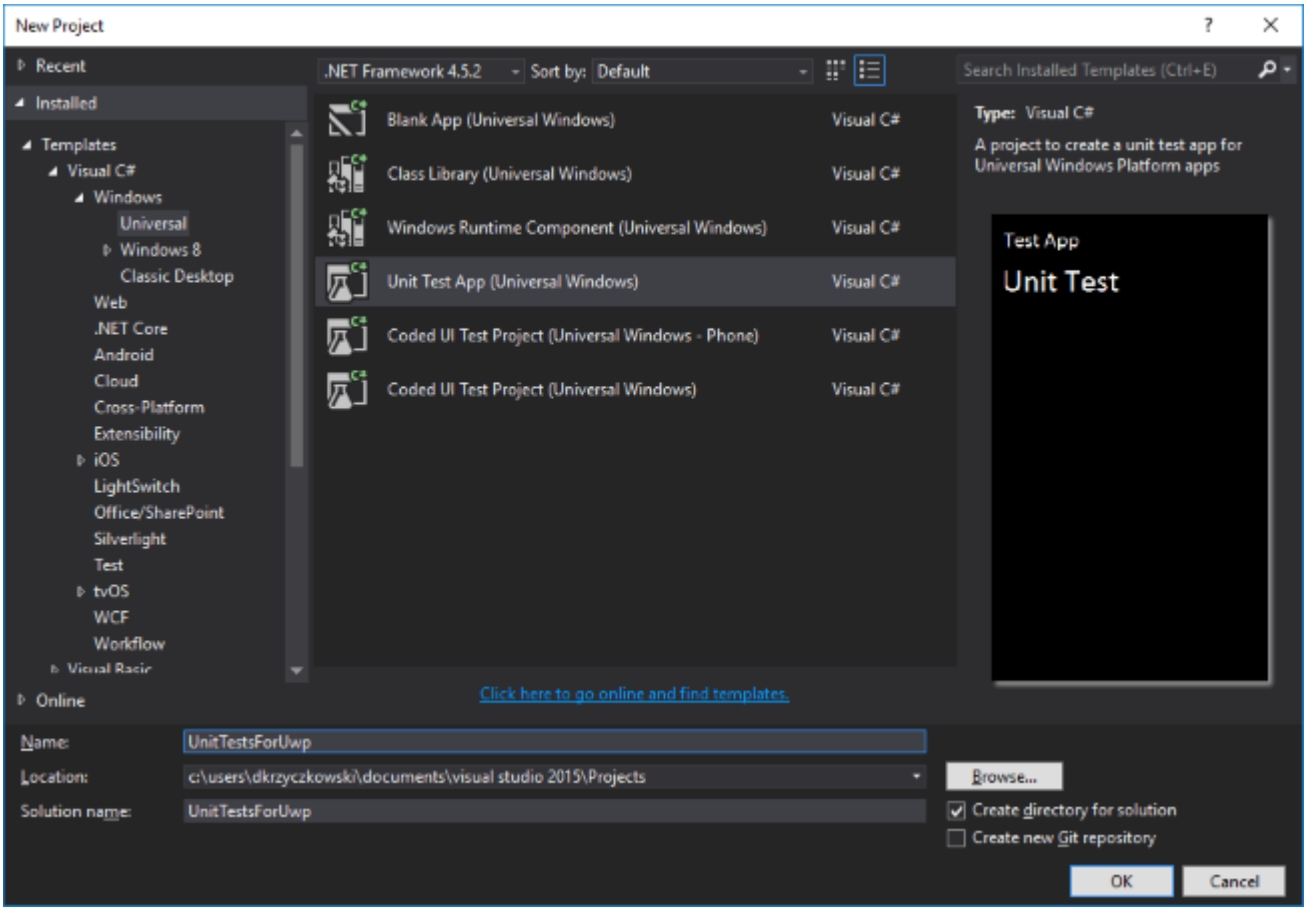
замечания

Вы можете узнать больше о xUnit Framewwork: https://xunit.github.io/docs/getting-started-uwp.html

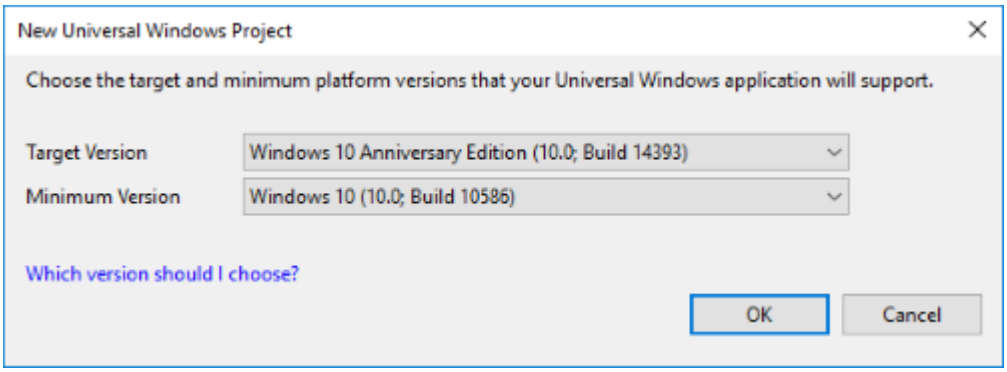
Examples

Настроить тестовое приложение

После того, как ваше приложение UWP готово к тестированию, вы должны добавить тестовое приложение к своему решению. Чтобы сделать это «правильно», нажмите на решение и выберите «Unit Test App (Universal Windows)»:



Как только вы добавите его в решение, для его настройки требуется еще несколько шагов. Вам будет предложено выбрать целевую и минимальную версию платформы:

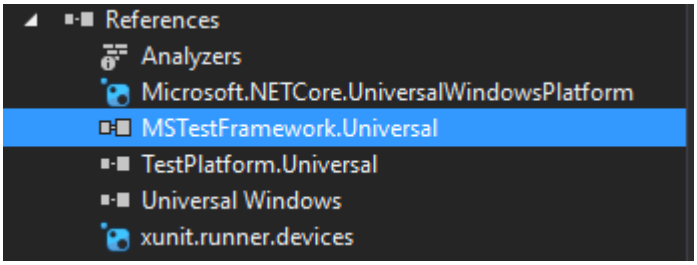


Выбрав их, откройте файл project.json и добавьте ниже зависимости:

```
"dependencies":
 {
  "Microsoft.NETCore.UniversalWindowsPlatform": "5.1.0",
  "xunit.runner.visualstudio": "2.1.0",
  "xunit": "2.1.0",
  "xunit.runner.devices": "2.1.0"
 }
```

Они используются для загрузки и добавления пакетов NuGet xUnit Framework для упрощения модульных тестов для приложения UWP.

Удалите ссылку под названием «MSTestFramework.Universal»:



Теперь откройте файл «UnitTest.cs». Измените это, чтобы выглядеть следующим образом:

```
using System;
using Xunit;

namespace UnitTestsForUwp
{
  public class UnitTest1
   {
       [Fact]
       public void TestMethod1()
        {
          Assert.Equal(4, 4);
        }

       [Theory]
       [InlineData(6)]
       public void TestMethod2(int value)
```

```
        {
          Assert.True(IsOdd(value));
        }

       bool IsOdd(int value)
        {
          return value % 2 == 1;
        }
     }
   }
}
```

Хорошо остановиться здесь, чтобы немного поговорить о атрибутах xUnit:

а. Факт-тесты, которые всегда верны. Они проверяют инвариантные условия.

б. Теория - тесты, которые справедливы только для определенного набора данных.

Теперь мы хотели бы подготовить приложение для отображения информации об испытаниях, но не только - хорошо иметь один хороший способ начать тесты. Для этого нам нужно внести небольшие изменения в файл «UnitTestApp.xaml». Откройте его и замените весь код следующим:

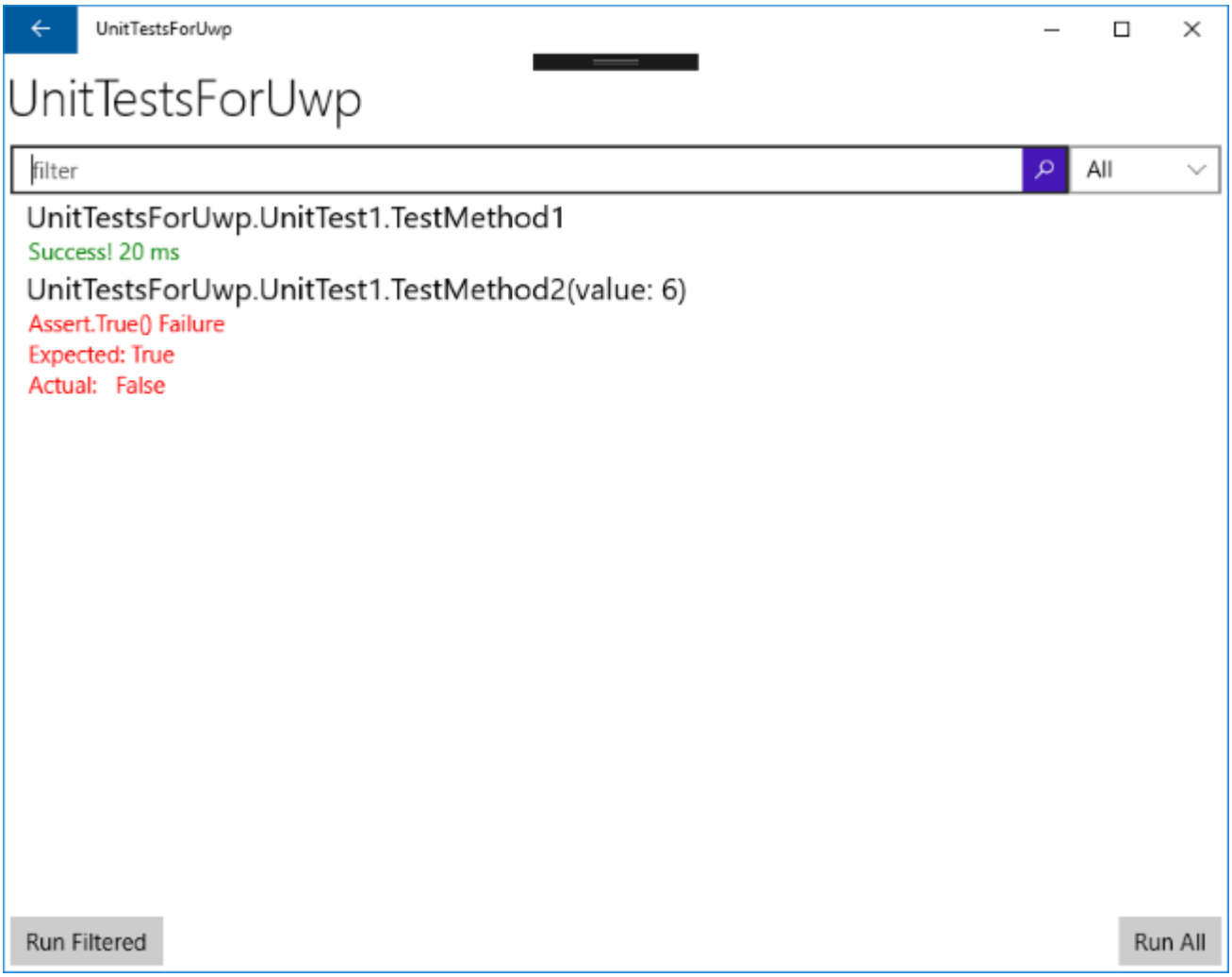
```
<ui:RunnerApplication
 x:Class="UnitTestsForUwp.App"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:local="using:UnitTestsForUwp"
 xmlns:ui = "using:Xunit.Runners.UI"
 RequestedTheme="Light">
</ui:RunnerApplication>
```

Помните, что «local» должен иметь то же имя, что и ваше пространство имен.

Теперь откройте «UnitTestApp.xaml.cs» и замените код ниже:

```
sealed partial class App : RunnerApplication
 {
   protected override void OnInitializeRunner()
    {
      AddTestAssembly(GetType().GetTypeInfo().Assembly);
      InitializeRunner();
    }
    partial void InitializeRunner();
  }
```

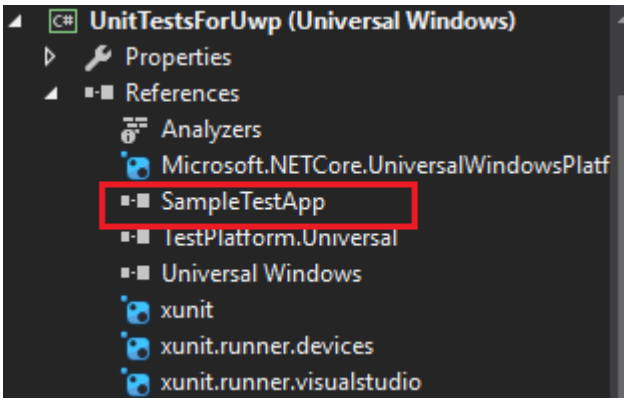
Это оно! Теперь перестройте проект и запускайте тестовое приложение. Как вы можете видеть ниже, у вас есть доступ ко всем вашим тестам, вы можете запустить их и проверить результаты:



Подключить тестовое приложение с целевым кодом приложения

Когда тестовое приложение будет готово, вы можете подключить его к коду, для которого вы хотите написать модульные тесты.

Либо у вас есть код в PCL, либо в проекте приложения UWP (я предполагаю, что вы применили шаблон MVVM) просто добавьте ссылку на него в проекте Test Application:



Теперь у вас есть доступ ко всему вашему коду из тестового приложения. Создайте необходимые тесты. Просто используйте «Факты» или «Теории».

Выполнить некоторые функции

Как только у вас есть все готово для написания ваших тестов единиц, стоит упомянуть насмешку. Существует новая структура под названием «SimpleStubs», которая позволяет создавать макеты на основе интерфейсов.

Простой случай из документации GitHub:

```
//Sample interface:
public interface IPhoneBook
 {
   long GetContactPhoneNumber(string firstName, string lastName);
   long MyNumber { get; set; }
   event EventHandler<long> PhoneNumberChanged;
 }

//Mocked interface:
var stub = new StubIPhoneBook().GetContactPhoneNumber((firstName, lastName) => 6041234567);
```

Подробнее об этом можно узнать здесь: https://github.com/Microsoft/SimpleStubs

Прочитайте Тестирование модулей для UWP онлайн: https://riptutorial.com/ru/uwp/topic/8634/тестирование-модулей-для-uwp

кредиты

S. No	Главы	Contributors
1	Начало работы с uwp	Community, dub stylee, Jerin, Josh, Pratyay
2	UWP Hello World	Almir Vuk, Arnaud Develay, Josh, user2950509
3	WebView	TableCreek
4	Адаптивный пользовательский интерфейс	Vincent
5	Жизненный цикл приложения	Daniel Krzyczkowski
6	Задачи UWP-фона	acedened, Askerad, Raben des Unbekannten, RamenChef, Romasz, TableCreek, Vincent
7	Изображений	acedened, AlexDrenea, Bart
8	Использование JavaScript в WebView	Vijay Nirmal
9	Как получить текущую DateTime в C ++ UWP	Mo0gles
10	навигация	Takarii, Vincent
11	Навигация по WebView	Andrey Ashikhmin
12	Настройки и данные приложения	khamitimur, Vincent
13	Отборочные имена файлов	Raben des Unbekannten
14	Преобразование размера	Dev-Systematix

	изображения и файла изображения обрезки в приложении Windows Universal	
15	Работа с файловой системой	Dale Chen
16	Ресурсы в UWP (StaticResource / ThemeResource) и ResourceDictionary	Ivan Carmenates García
17	Связывание vs x:Bind	Alias Varghese, Anthony Russell, Askerad, Raben des Unbekannten, Vincent
18	Семейство устройств	Josh, M. Pipal, Vincent
19	Тематические ресурсы	Askerad, Bart, Raamakrishnan A.
20	Тестирование модулей для UWP	Daniel Krzyczkowski