Windows 10

Лабораторный практикум

Программа "Hello World" для универсальной платформы Windows

Октябрь 2015 года

Microsoft

Общие сведения

В Windows 10 впервые появилась универсальная платформа Windows (UWP), которая представляет собой дальнейшее развитие модели Windows Runtime и включает её в ядро ОС Windows 10. Как часть ядра, UWP сейчас предоставляет собой стандартную платформу для приложений, доступную на каждом устройстве, которое запускается под Windows 10. В результате приложения, работающие на UWP на устройстве определенного класса, могут вызывать специфические для данного семейства устройств API, в дополнение к API WinRT, которые являются общими для всех устройств. Вы можете с помощью UWP создать один пакет приложения, который может быть установлен на широкий ассортимент устройств.

В этой работе вы будете использовать инструменты разработки для универсальной платформы Windows (Universal Windows App Development Tools) для создания приложения "Hello World", которое будет запускаться на всех устройствах под Windows 10. Ваше приложение будет отображать информацию об устройстве, на котором оно выполняется, включая семейство устройства и размер текущего окна приложения. Вы также создадите приложение "Hello World" в среде Blend и используете возможности Blend для генерации выборки данных.

Цели

Эта работа научит вас:

- Создавать UWP приложение из пустого шаблона.
- Отображать приветствие в своём приложении
- Находить и отображать данные о семействе устройства
- Динамически отображать размер окна приложения
- Запускать приложение на локальном компьютере
- Запускать эмулятор мобильной платформы
- Запускать приложение на устройствах "интернета вещей"
- Генерировать выборку данных в Blend

Системные требования

Чтобы выполнить этот курс, необходимо обладать следующим набором программных инструментов:

- Microsoft Windows 10
- Microsoft Visual Studio 2015

Дополнительные задачи

Если вы захотите проходить дополнительные задачи в этом курсе, вам понадобятся:

- Устройство Windows 10 Mobile или эмулятор
- Устройство "интернета вещей", запускающееся под Windows 10 (Raspberry Pi 2)
- Дисплей, который соединяется с устройством "интернета вещей"

Настройка

Вы должны осуществить следующие шаги для подготовки своего компьютера для этого курса:

1. Установите Microsoft Windows 10.
2. Установите Microsoft Visual Studio 2015. Выберите пользовательскую установку и убедитесь, что в списке дополнительных функций выбраны инструменты разработки для приложений Windows.
3. Дополнительно: Установите эмулятор Windows 10 Mobile.
4. Дополнительно: Установите Windows 10 на устройство "интернета вещей".

Упражнения

Эта практическая работа включает следующие упражнения:

1. Начало работы с универсальной платформой Windows
2. Программа "Hello World" на устройствах под Windows
3. Программа "Hello World" в среде Blend

Расчётное время для завершения работы: **от 30 до 45 минут**.

Упражнение 1: Начало работы с универсальной платформой Windows

Если у вас установлены универсальные инструменты разработки приложений Windows, то в Visual Studio будет доступн шаблон для создания приложений UWP. В ходе этого упражнения вы создадите проект из пустого шаблона приложения.

Задача 1 – Создаем пустое приложение под Windows

Мы начнём с создания проекта из шаблона приложения.

1. В новой версии Visual Studio 2015 выберите File (Файл) -> New (Новый) -> Project (Проект), чтобы открыть диалоговое окно New Project (Новый проект). Далее выберите Installed (Установленные) > Templates (Шаблоны) > Visual C# > Windows > Universal, а затем выберите шаблон Blank App приложения Universal Windows.

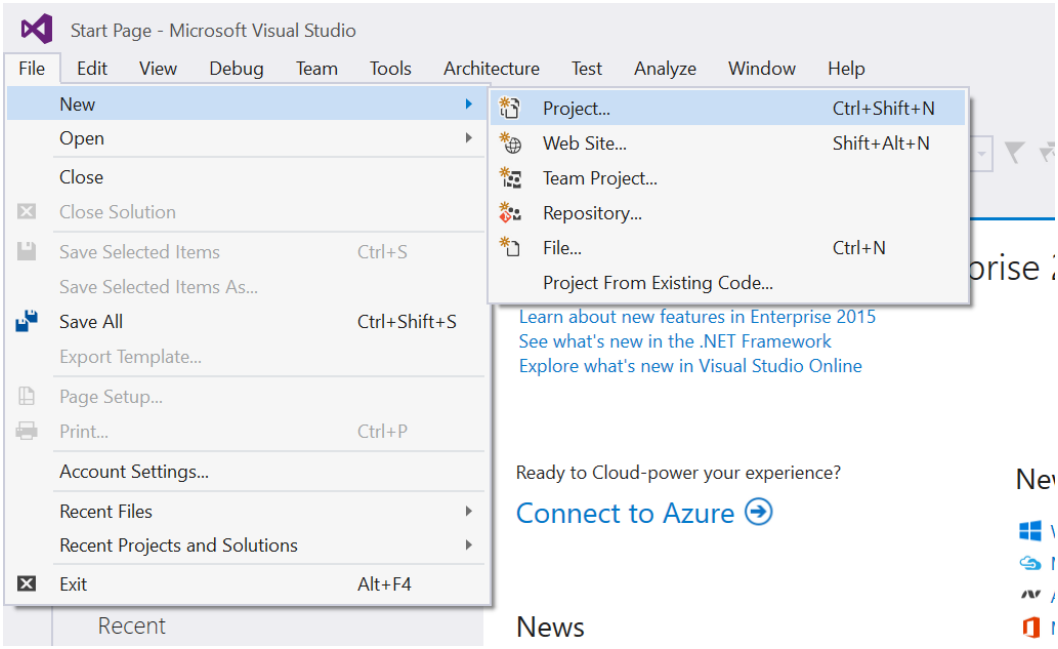

Рисунок 1

Открытие диалогового окна New Project (Новый проект) в Visual Studio 2015.

2. Назовите свой проект **HelloUWP** и выберите местоположение в файловой системе, где вы храните работы этого практикума. Мы создали папку на диске **C** и переименовали ее в **HOL**, именно папку с этим названием вы будете видеть на скриншотах в ходе всего практического курса.

3. Выберите опции **Create new solution (Создать новое решение)** и **Create directory for solution (Создать папку для решения)**. Вы можете снять галочку с опции **Add to source**

control (Добавить контроль исходного кода), если вы не хотите хранить свой код в каком-либо репозитории. Нажмите **ОК** для создания проекта.

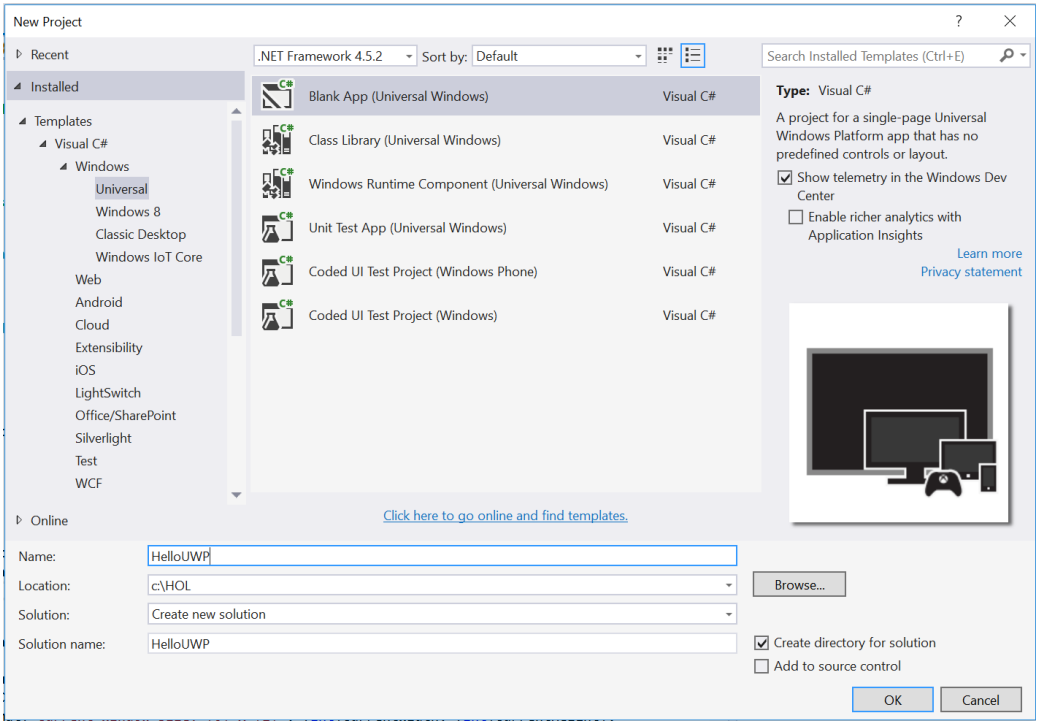

Рисунок 2

Создание нового проекта приложения в Visual Studio 2015.

4. Установите значение Debug (Отладка) для параметра Solution Configuration (Конфигурации решения), и значение x86 для параметра Solution Platform (Платформа решения). Выберите локальный компьютер в выпадающем списке Debug Target (Цели отладки) возле кнопки Start Debugging (Запустить отладку).

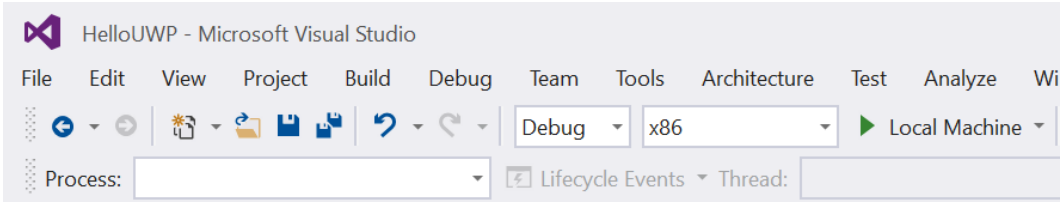

Рисунок 3

Конфигурация приложение для запуска на локальном компьютере.

Примечание: ▶ Кнопка Start Debugging (Запуск отладки).

5. Нажмите кнопку Start Debugging (Запуск отладки), чтобы создать и запустить своё приложение. Вы увидите пустое окно приложения со счётчиком скорости перерисовки экрана, который показывается по умолчанию в режиме отладки.

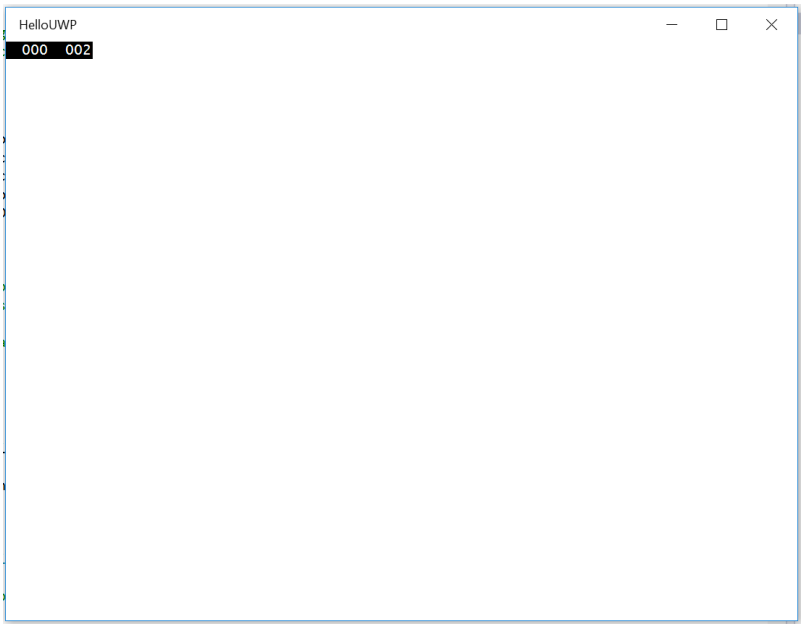

Рисунок 4

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

Примечание: Счётчик скорости обновления кадра является инструментом отладки, который помогает следить за производительностью вашего приложения. Он полезен для приложений, которые требуют интенсивной обработки графики, но не используется для простых приложений, которые вы будете создавать в рамках практикума.

В шаблоне приложения директива препроцессора для активации или отключения счётчика скорости кадра находится в файле **App.xaml.cs**. Счётчик скорости обновления кадра может перекрывать или скрывать содержание приложения, если вы оставите его включенным. Поэтому мы отключим его.

6. Вернитесь в Visual Studio и отключите отладку.

7. Откройте App.xaml.cs. Для отключения счётчика скорости кадра в директиве препроцессора **#if DEBUG**, установите значение **this.DebugSettings.EnableFrameRateCounter** в **false**.

```C#
#if DEBUG
    if (System.Diagnostics.Debugger.IsAttached)
    {
        this.DebugSettings.EnableFrameRateCounter = false;
    }
#endif
```

8. Создайте и запустите своё приложение повторно. В этот раз вы увидите пустое окно приложения без счётчика скорости обновления кадров.

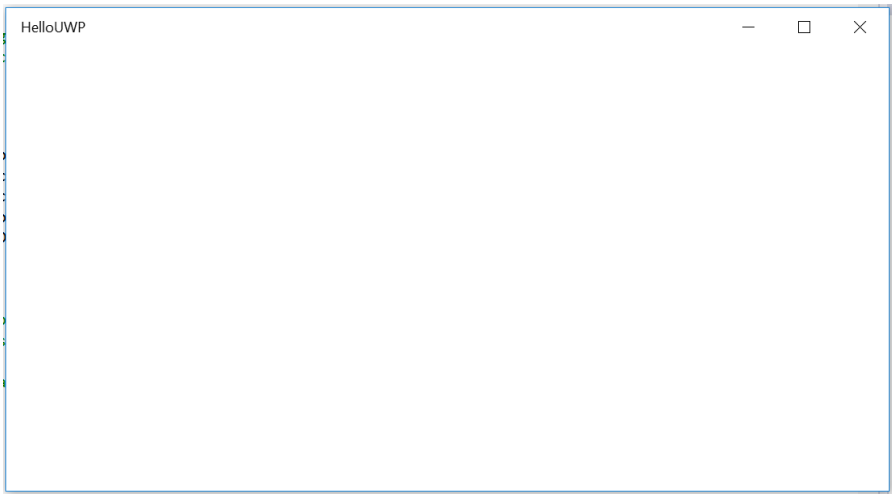

Рисунок 5

Пустое приложение после отключения счётчика скорости обновления кадров.

9. Отключите отладку и вернитесь в Visual Studio.

Задача 2 – Изучаем шаблон

Шаблон пустого приложения обеспечивает базовую структуру, которая вам будет нужна при создании приложения UWP. Давайте рассмотрим, как он устроен.

1. **App.xaml.cs** – точка входа для вашего приложения, этот файл содержит методы обработки активации и приостановления работы приложения. В этом шаблоне конструктор запускает инструмент Application Insights, вызывает функцию InitializeComponent() и изучает список отложенных событий. В соответствующем файле **App.xaml**, вы можете объявить ресурсы, которые будут совместно использованы в приложении.

   Функция **OnLaunched** в App.xaml.cs настраивает содержимое страницы (Frame) как навигационный контекст и подключает его к текущему окну (Window). Вы можете добавить код для загрузки состояния предыдущего исполнения приложения (это важно, чтобы приложение могло восстановить своё состояние после приостановки). Если состояние не было сохранено, произойдет переход на страницу **MainPage**.

   Вы также можете заметить, что функция **OnLaunched()** включает код для активации счётчика скорости кадра при отладке, условно включенного в директиву препроцессора `#if DEBUG` . Дополнительную информацию о некоторых параметрах отладки вы можете получить по ссылке: https://msdn.microsoft.com/en-us/library/windows.ui.xaml.debugsettings.aspx.

   Функция **Window.Current.Activate()** является важной частью процесса активации и требуется для всех сценариев активации приложения, включая второстепенные окна.

2. **MainPage.xaml** и **MainPage.xaml.cs** совместно определяют вид страницы **MainPage**, унаследованной от базового класса **Page**. Вы можете добавить элементы

пользовательского интерфейса в **MainPage.xaml** и обработчики логики и событий в код **MainPage.xaml.cs**. Вы сможете больше узнать о MainPage в следующем упражнении.

3. Папка **Assets (Ресурсы)** содержит ресурсы по умолчанию для приложения. В шаблоне пустого приложения представлен минимальный набор ресурсов. Если вы захотите задать собственные или дополнительные ресурсы, вы можете добавить их в эту папку, кликнув правой кнопкой мыши на имени папки в Solution Explorer и выбрав **Add (Добавить) > Existing Item (Существующий компонент)**. Как только ресурсы были добавлены, вы можете посетить манифест приложения, чтобы зарегистрировать свои новые ресурсы.

4. Файл **Package.appxmanifest** описывает свойства вашего приложения. В этот файл вы можете добавить разрешение на использование определенных функций и расширений – например, функцию доступа устройства к микрофону. В нем также содержится ссылка на экран заставки, "плитку" по умолчанию и иконка приложения. **Package.appxmanifest** открывается по умолчанию в **Manifest editor (Редактор манифеста)**, который позволяет редактировать все полезные свойства приложения в режиме диплога. На вкладке **Visual Assets (Визуальные ресурсы)**, например, редактор проектов указывает, какие ресурсы логотипов рекомендованы к использованию. Вы также можете просматривать манифест приложений в виде XML файла: чтобы сделать это, кликните правой кнопкой мыши на файле в Solution Explorer и выберите **View Code (Просмотр кода)**.

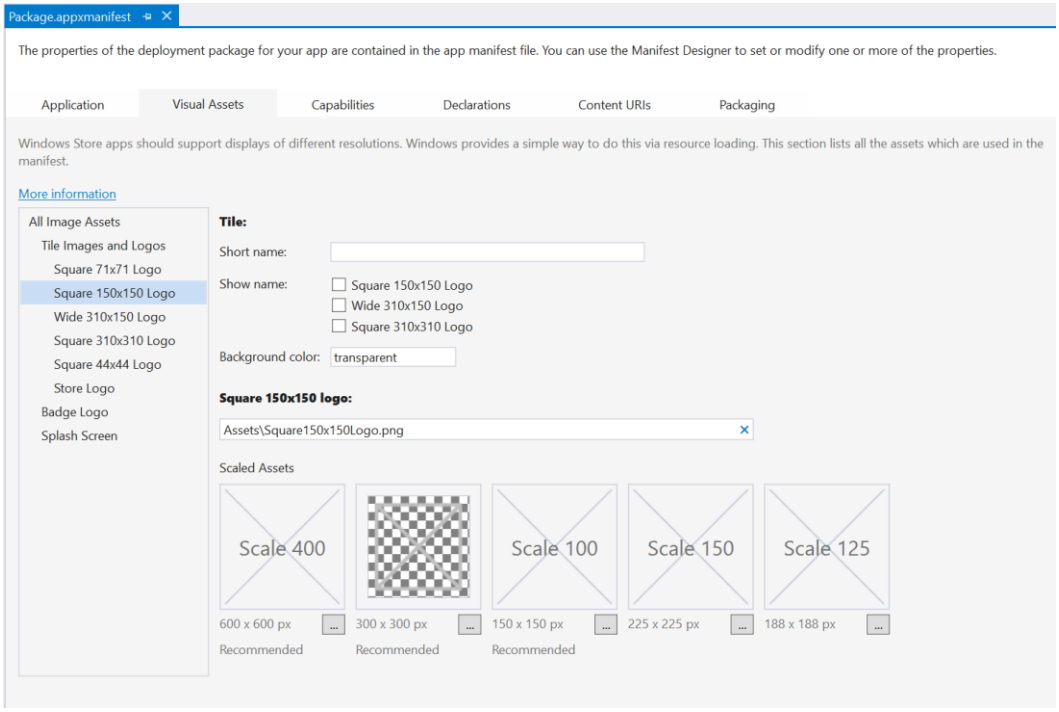

Рисунок 6

В манифесте приложений указывается, какие иконки рекомендованы к использованию.

5. Папка **References (Ссылки)** содержит список SDK, библиотек и компонентов, которые используются в вашем приложении. Если вы хотите добавить ссылку, кликните правой кнопкой мыши на папке References (Ссылки) и выберите **Add Reference (Добавить Ссылку)**, чтобы открыть **Reference Manager (Менеджер ссылок)**.

6. Файл **project.json** в корневом каталоге вашего проекта является новым для платформы UWP и упрощает управление пакетами NuGet (пакетами внешних подключаемых библиотек). Как правило, лучше не редактировать этот файл напрямую, а использовать средства менеджера пакетов NuGet для просмотра и управления пакетами для использования в своём проекте. Дополнительную информацию см. по ссылке http://docs.nuget.org/consume/ProjectJson-Intro.

7. Файл **Properties** позволяет вам управлять событиями сборок и конфигурацией отладки. По мере дальнейшего развития платформы Windows 10, вам может потребоваться использовать вкладку **Application (Приложение)**, чтобы обновить оптимальную версию и минимальную версию платформы UWP и Windows 10, которые ваше приложение будет поддерживать.

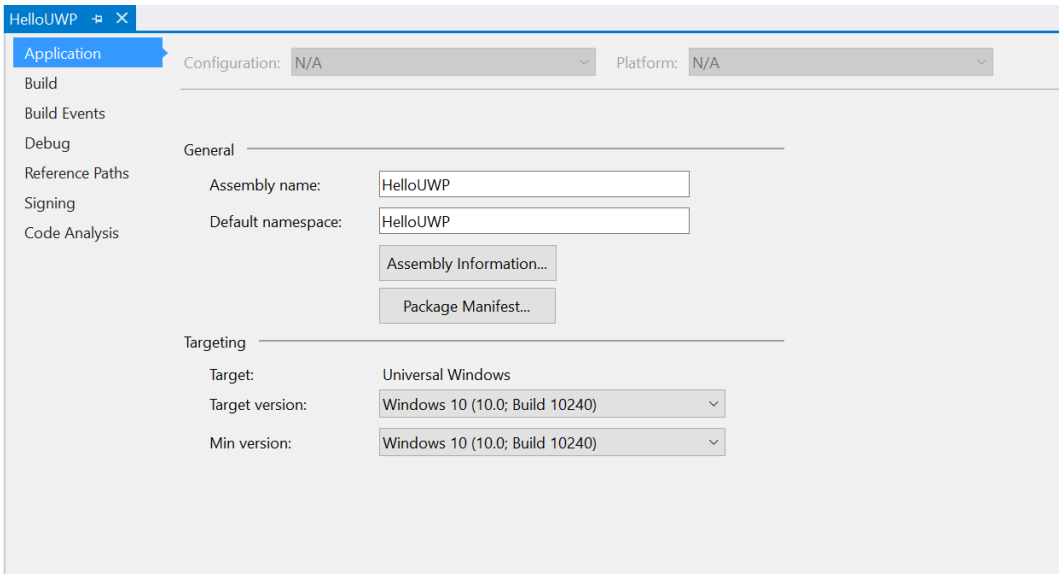

Рисунок 7

Изменение поддерживаемой целевой и минимальной версии Windows 10 в свойствах вашего приложения.

Упражнение 2: Программа "Hello World" на устройствах под Windows

Приложения Windows сейчас являются универсальными для всех устройств, что означает, что они могут работать в полноэкранном режиме на мобильных телефонах и планшетах, а также в оконном режиме на персональных компьютерах. В этом упражнении вы создадите приложение "Hello World", которое отображает приветствие и информацию об устройствах под Windows 10, включая настольные компьютеры, планшеты, смартфоны и устройства "интернета вещей".

Задача 1 – Отображение приветствия

Первая задача состоит в том, чтобы открыть решение HelloUWP, которое вы создали в предыдущем упражнении.

1. Откройте директорию, в которой вы сохранили своё приложение **HelloUWP** в Упражнении 1. Откройте файл **HelloUWP.sln** в Visual Studio 2015. Также вы можете открыть решение, используя диалоговое окно **Open Project (Открыть Проект)** Visual Studio.

2. Откройте **MainPage.xaml** в Solution Explorer. Добавьте элемент **StackPanel**, содержащий текстовый блок **TextBlock** к существующему файлу разметки.

XAML

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel>
        <TextBlock />
    </StackPanel>
</Grid>
```

Примечание: Элемент **StackPanel** является контейнерным элементом управления, который расставляет дочерние элементы вертикально или горизонтально. Есть некоторое количество других контейнеров, которые могут помочь в компоновке вашего пользовательского интерфейса, включая **Grid (Сетка)**, **Canvas (Холст)** и **RelativePanel (Дополнительная панель)**. Мы опишем элементы Grid (Сетка) и RelativePanel (Дополнительная панель) позже в рамках курса **Создание адаптивного пользовательского интерфейса**.

3. Добавление текста и дизайна в ваш блок TextBlock.

XAML

```
<TextBlock Text="Hello UWP World!" FontSize="24" FontWeight="Light"
Margin="12" />
```

Примечание: Вы можете заметить, что во всех практикумах значения полей, размера и положения всегда кратны 4. Приложения UWP используют эффективные пиксели на 4x4-пиксельной сетке, чтобы гарантировать высокое качество визуального опыта в устройствах, использующих экраны с различными разрешениями и масштабами. Чтобы гарантировать правильный вид для вашего приложения, размещайте элементы пользовательского интерфейса в 4x4-пиксельной сетке. Эта сетка не применяется к тексту, так что вы можете продолжить использовать любой размер шрифта, который пожелаете.

Более подробную информацию о эффективных пикселях и соответствующем дизайне приложений см. по ссылке https://msdn.microsoft.com/en-us/library/windows/apps/Dn958435.aspx

4. Соберите и запустите своё приложение на **локальном компьютере**.

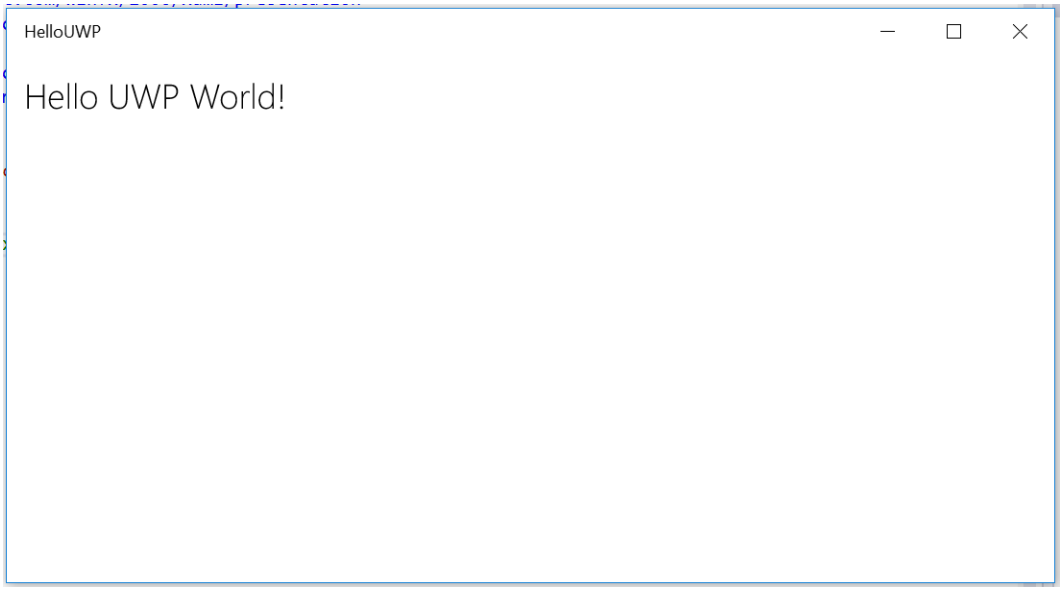

Рисунок 8

Приложение HelloUWP отображает приветствие.

5. Отключите отладку и вернитесь в Visual Studio. Позднее в этом упражнении вы научитесь запускать приложение в мобильном эмуляторе и на устройствах «интернета вещей». Необходимо добавить дополнительные элементы управления для отображения информации, которая будет указывать тип устройства и размеры экрана.

Задача 2 – Определение семейства устройства

Сейчас, когда одно приложение UWP может выполняться на всех устройствах под Windows 10, было бы полезно иметь возможность определять отдельные семейства устройств, которые совместно используют специфические функции API. Например, вы можете задать некоторые функции, которые будут доступны только на мобильных устройствах, и скрыть их на настольной версии приложения. Для этого вводят понятие **семейства устройств**, т.е. набора устройств, для которых имеются те или иные разновидности API. В этой задаче вы обнаружите и покажете семейство устройства, на котором выполняется ваше приложение.

1. Откройте **MainPage.xaml.cs**. Добавьте поле, чтобы запомнить информацию о семействе устройств.

C#

```
public string DeviceFamily = "Device Family " +
                             AnalyticsInfo.VersionInfo.DeviceFamily;

public MainPage()
{
    this.InitializeComponent();
}
```

Примечание: С приходом Windows 10 меняется взгляд на таргетирование приложений. Согласно новой концептуальной модели, приложение ориентируется на группу родственных устройств, называемых семейством устройств. Семейство устройств – набор API, собранных вместе и наделённых названием и номером версии. Ваше приложение может запускаться на множестве семейств устройств через адаптивный код и иметь специфические уникальные функции для каждого семейства устройств.

Более подробную информацию о семействах устройств в приложениях UWP см. по ссылке https://msdn.microsoft.com/en-us/library/windows/apps/dn894631.aspx

2. Добавьте пространство имен **Windows.System.Profile** в файле MainPage.

C#

```
using Windows.System.Profile;
```

3. В MainPage.xaml свяжите **TextBlock** с полем **DeviceFamily**.

Примечание: Связывание данных обеспечивает простой способ взаимодействия вашего UI с данными. Отображение данных отделяется от бизнес-логики приложения, а связывание позволяет осуществлять привязку свойств элементов UI с полями данных. Связывание может использоваться для одноразового отображения данных (One Time), обновления UI при изменении данных (One Way) или захвата данных, вводимых пользователями (Two Way).

{x:Bind} является новым расширением разметки в Windows 10 UWP и предлагает компилируемую альтернативу классическому связыванию {Binding}. Более подробную информации о x:Bind см. по ссылке https://msdn.microsoft.com/en-us/library/windows/apps/Mt204783.aspx

XAML

```
<StackPanel>
    <TextBlock Text="Hello UWP World!" FontSize="24" FontWeight="Light"
Margin="12" />
    <TextBlock Text="{x:Bind DeviceFamily}" Margin="12,0,0,0" />
</StackPanel>
```

Примечание: StackPanel по умолчанию располагает элементы вертикально. Вы можете указать горизонтальную ориентацию с помощью атрибута Orientation=”Horizontal”.

4. Соберите и запустите своё приложение на **локальном компьютере**. Вы увидите отображение данных о семействе устройства под приветствием.

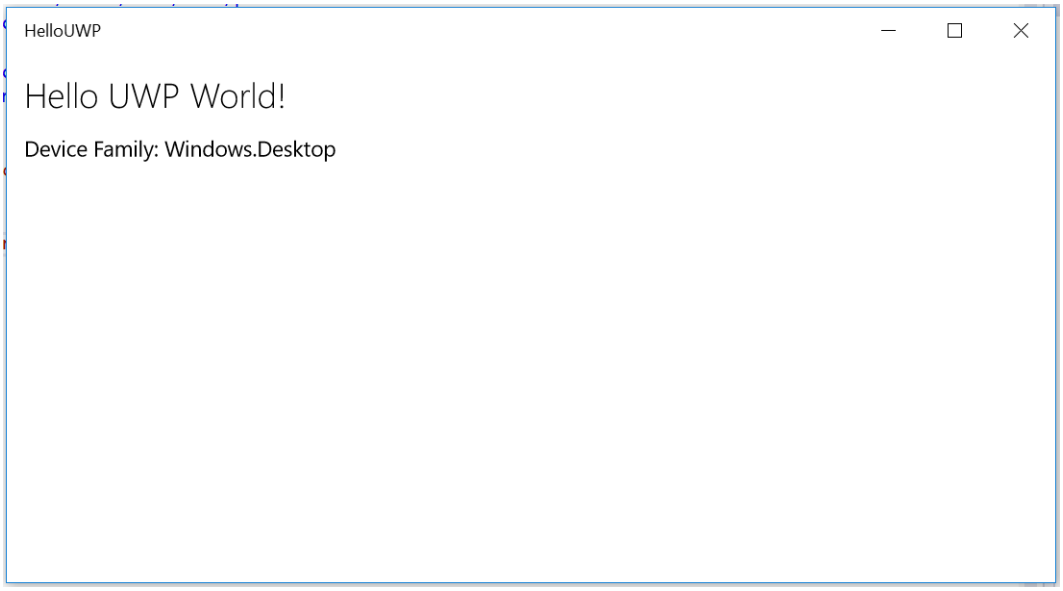

Рисунок 9

Приложение HelloUWP отображает семейство устройств для устройства, на котором оно выполняется.

> **Примечание:** Если вы занимаетесь разработкой в настольной среде и используете локальный компьютер, ваше семейство устройств отобразится как Windows.Desktop. Список других семейств устройств, доступных в Universal Device Family включает Mobile, Xbox, IoT, IoT headless и другие.

5. Отключите отладку и вернитесь в Visual Studio.

Задача 3 – Динамическое отображение размеров окна приложения

Очередное свойство, которое любопытно сравнивать на разных устройствах – размер окна приложения, который выражен в эффективных пикселях. Универсальные приложения Windows могут выполняться в полноэкранном режиме на мобильных устройствах и планшетах, а также в оконном режиме на рабочих станциях. В этой задаче вы создадите код для считывания данных и отображения размерности окна приложения. Данные будут обновляться, когда вы измените размер окна приложения.

1. Добавьте блок **TextBlock** в элемент управления **StackPanel** в файле MainPage.xaml и привяжите его к свойству **Dimensions (Размеры)**. Вы создадите свойство **Dimensions (Размеры)** в следующих шагах.

XAML

```
<StackPanel>
    <TextBlock Text="Hello UWP World!" FontSize="24" FontWeight="Light"
Margin="12" />
    <TextBlock Text="{x:Bind DeviceFamily}" Margin="12,12,0,0" />
```

```
    <TextBlock Text="{x:Bind Dimensions, Mode=TwoWay}" Margin="12,0,0,0" />
</StackPanel>
```

1. Откройте **MainPage.xaml.cs** и создайте свойства для размеров, текущей ширины и текущей высоты.

C#

```
public string DeviceFamily = "Device Family " +
                             AnalyticsInfo.VersionInfo.DeviceFamily;

public string Dimensions { get; set; } = "Initial Value";
private double _currentWidth;
private double _currentHeight;
```

2. Добавьте обработчик события **MainPage_SizeChanged** в код ниже. Когда событие **SizeChanged** запустится, этот обработчик определит новую ширину и высоту окна и округлит их до целочисленных значений для отображения.

C#

```
public MainPage()
{
    this.InitializeComponent();
    this.SizeChanged += MainPage_SizeChanged;
}

private void MainPage_SizeChanged(object sender, SizeChangedEventArgs e)
{
    _currentWidth = Window.Current.Bounds.Width;
    _currentHeight = Window.Current.Bounds.Height;
    Dimensions = string.Format("Current Window Size: {0} x {1}",
(int)_currentWidth, (int)_currentHeight);
}
```

3. Чтобы ваше скомпилированная привязка работала со свойством **Dimensions (Размеры)**, вы должны вызвать событие **PropertyChanged**. Используйте интерфейс **INotifyPropertyChanged**.

C#

```
public sealed partial class MainPage : Page, INotifyPropertyChanged
{
    public string DeviceFamily = "Device Family " +
                             AnalyticsInfo.VersionInfo.DeviceFamily;

    public event PropertyChangedEventHandler PropertyChanged;

    public string Dimensions { get; set; } = "Initial Value";
    private double _currentWidth;
    private double _currentHeight;


    public MainPage()
```

```
    {
        this.InitializeComponent();
        this.SizeChanged += MainPage_SizeChanged;
    }

    private void MainPage_SizeChanged(object sender,SizeChangedEventArgs e)
    {
        _currentWidth = Window.Current.Bounds.Width;
        _currentHeight = Window.Current.Bounds.Height;
        Dimensions = string.Format("Current Window Size: {0} x {1}",
(int)_currentWidth, (int)_currentHeight);

        if (PropertyChanged != null)
        {
            PropertyChanged(this,
                            new
PropertyChangedEventArgs(nameof(Dimensions)));
        }
    }
```

4. Добавьте пространство имен **System.ComponentModel** в код MainPage для завершения реализации использования интерфейса INotifyPropertyChanged.

C#

```
using System.ComponentModel;
```

5. Скомпилируйте и запустите своё приложение. Текстовый блок **Dimensions (Размеры)** отобразит исходный размер окна и обновится при изменении размера окна.

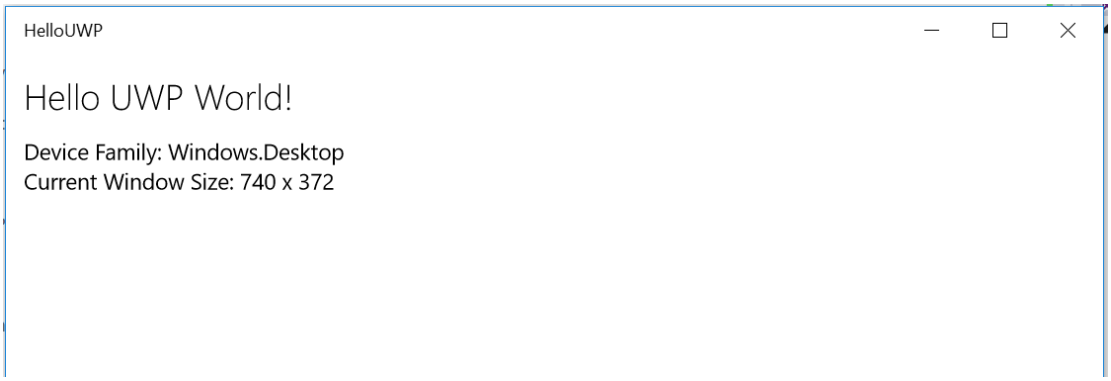

Рисунок 10

Размер обновляется, поскольку размер окна изменён.

6. Остановите отладку и вернитесь в Visual Studio.

Задача 4 – Запуск эмулятора мобильной платформы (опционально)

Приложение HelloUWP, которое вы создали в предыдущей задаче, уже настроено для запуска на любом устройстве Windows 10. До появления UWP приложений разработчикам приходилось бы создавать отдельный проект для приложения, работающего на мобильных

устройствах Windows Phone. Преимущество UWP состоит в том, что один проект будет работать на всех устройствах под Windows 10. В этой задаче мы будем использовать эмулятор мобильной платформы Windows 10 Mobile, чтобы протестировать работу приложения HelloUWP на мобильном устройстве.

> **Примечание:** Эмулятор Microsoft мобильной платформы на Windows 10 является частью SDK для Windows 10, и может быть установлен во время процесса установки Visual Studio 2015. Для работы эмулятора требуется Windows 8.1 (x64) Professional edition и выше или Windows 10 Pro или Enterprise (x64). Кроме того, вам понадобится процессор, который поддерживает технологию Client Hyper-V и Second Level Address Translation (SLAT).
>
> Посетите страницу системных требований для Visual Studio 2015 для получения дополнительной информации.

1. Измените значение настройки **Debug Target (Цель Отладки)**, чтобы использовать один из эмуляторов, предоставляемых Microsoft Emulator for Windows 10 Mobile. Мы решили использовать опцию **Mobile Emulator 10.0.10240.0 720p 5 inch 1GB**.

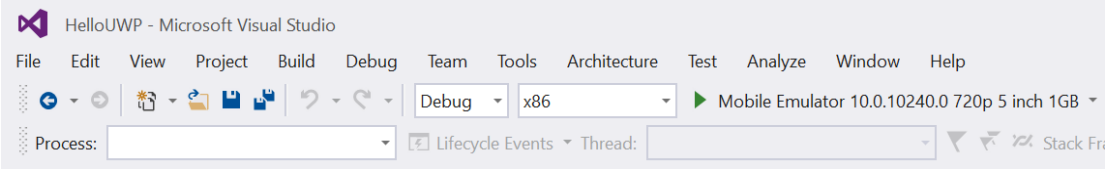

Рисунок 11

В качестве цели отладки установлен эмулятор мобильной платформы.

2. Скомпилируйте и запустите своё приложение. Приложение HelloUWP отобразится на вашем мобильном устройстве со значением семейства устройств Windows.Mobile. Размер окна будет отражать размеры экрана мобильного устройства (за исключением строки состояния), потому что приложения мобильного всегда работают в полноэкранном режиме.

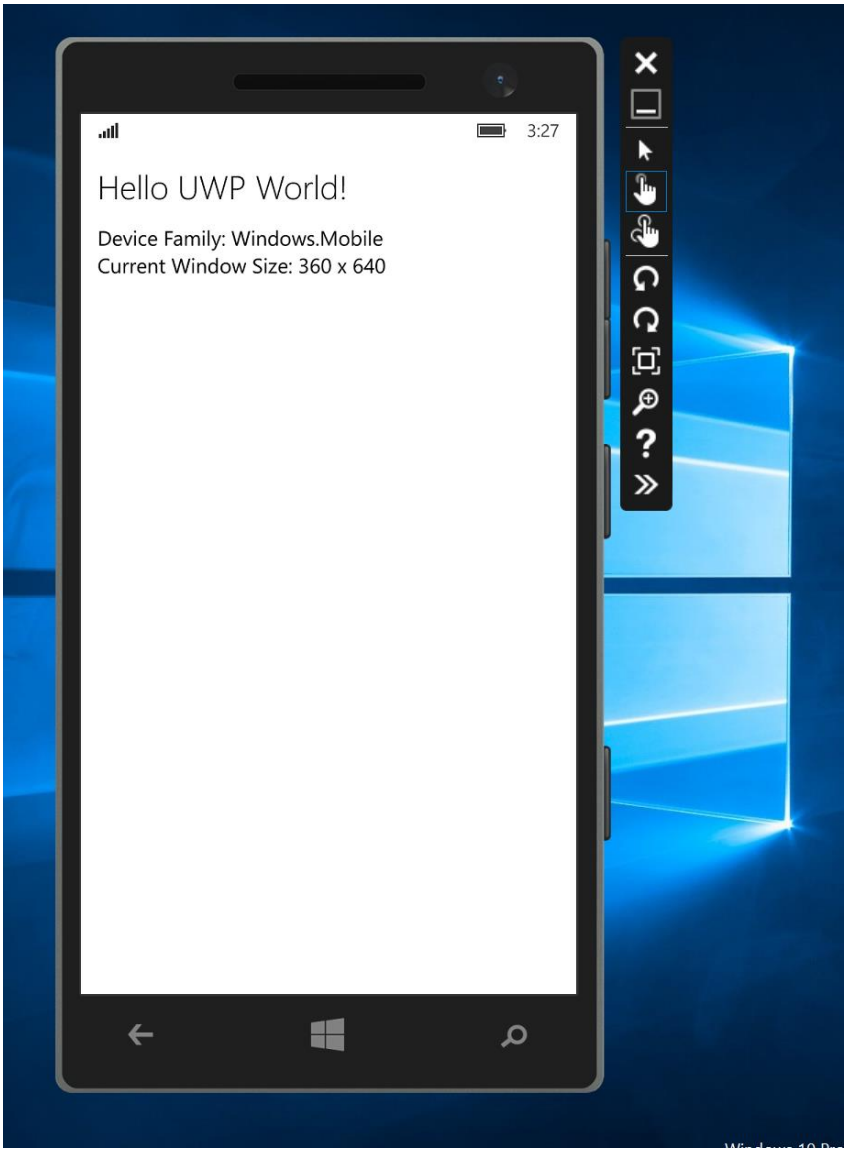

Рисунок 12
HelloUWP запущена в эмуляторе мобильной платформы.

3. Остановите отладку и вернитесь в Visual Studio.

Задача 5 – Запуск приложения на устройствах "интернета вещей" (дополнительно)

Семейство универсальных устройств Windows 10 включает устройства "интернета вещей" (IoT) в дополнение к настольным компьютерам, мобильным устройствам и Xbox. Устройства IoT обычно включают датчики и средства подключения для записи и обмена данными с другими устройствами. Вы можете выполнить эту задачу для запуска своего приложения HelloUWP на устройстве IoT, если вы имеете его в своем распоряжении. Для этой задачи мы будем использовать Raspberry Pi 2, работающее под Windows 10. Если вы не имеете доступа к устройству IoT, вы можете пропустить эту задачу.

Примечание: Пройдите по ссылке **http://www.windowsondevices.com/** для загрузки инструкций и инструментов для установки Windows 10 на устройство IoT. Список поддерживаемых устройств включает Raspberry Pi, Minnowboard Max, Galileo и Arduino.

Семейство универсальных устройств также включает семейство IoT устройств без собственных средств управления. Подобные устройства IoT работают без графического интерфейса пользователя. Наше приложение HelloUWP отображает информацию визуально, так что мы будем работать с семейством обычных IoT устройств вместо семейства IoT устройств без графического интерфейса (headless).

1. Убедитесь, что ваше устройство IoT имеет питание, и загрузите его под Windows 10. Подключите устройство к монитору.

2. Подключите устройство IoT к своей локальной сети. Вы можете подключиться напрямую посредством Ethernet или Wi-Fi или использовать совместный доступ в Интернет (ICS), чтобы подключиться через свой компьютер разработчика.

Примечание: Дополнительную информацию по подключению своего устройства IoT к своей локальной сети см. по ссылке https://ms-iot.github.io/content/en-US/win10/ConnectToDevice.htm.

3. Задайте локальный IP-адрес для своего устройства IoT. Raspberry Pi 2, запускающееся под Windows 10, по умолчанию для устройств IoT отображает имя устройства и IP-адрес на начальном экране.

4. Используйте PowerShell, чтобы подключить и сконфигурировать своё устройство IoT под Windows 10, как описано по ссылке: http://ms-iot.github.io/content/en-US/win10/samples/PowerShell.htm Во многих случаях дополнительные шаги конфигурации не нужны.

5. Откройте проект HelloUWP в Visual Studio на своем рабочем компьютере. Настройте Solution Platform (Платформа решения) на **ARM**.

6. Дважды кликните по **Properties (Свойства)** проекта в Solution Explorer. Вы также можете правой кнопкой мыши кликнуть на названии проекта и выбрать пункт **Properties (Свойства)** из меню.

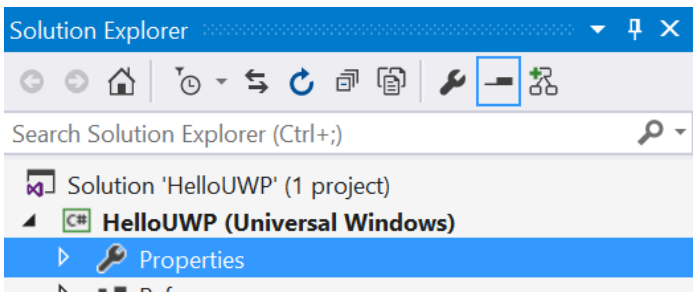

Рисунок 13

Откройте свойства проекта в Solution Explorer.

1. Перейдите на вкладку свойств **Debug (Отладка)**. Используйте выпадающий список **Target device (Целевое устройство)** для выбора **Remote Machine (Удалённый компьютер)** в качестве цели отладки.

2. Отмените выбор опции **Use authentication (Аутентификация при использовании)**.

3. Выведите имя устройства IoT или локальный IP-адрес в поле **Remote machine (Удалённый компьютер)**.

4. Сохраните файл свойств HelloUWP.

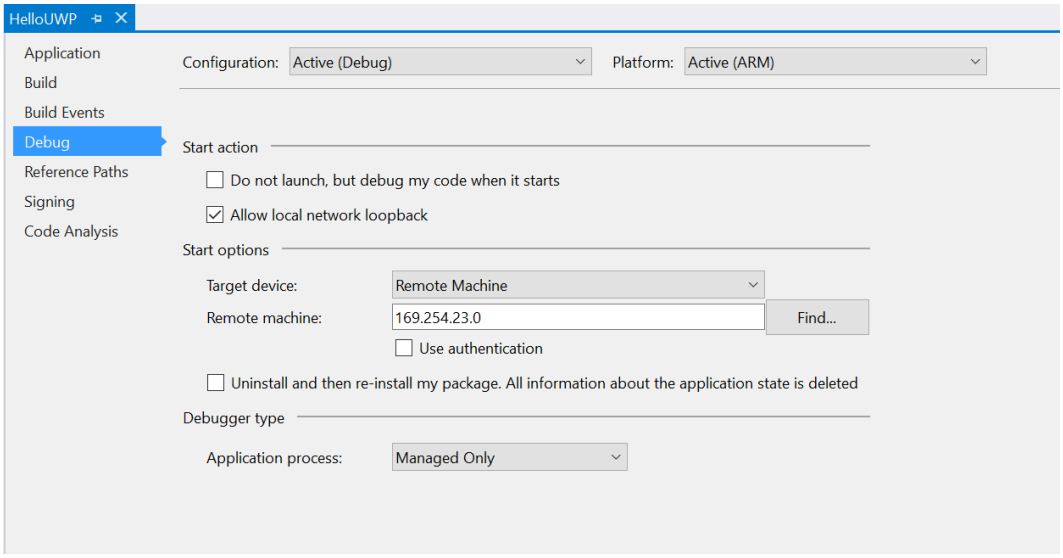

Рисунок 14

Сконфигурируйте свойства проекта для запуска на удалённом компьютере в локальной сети.

5. Скомпилируйте и запустите своё приложение. Вы увидите, что оно запустилось на дисплее, подключённом к вашему устройству IoT. Семейство Устройств будет отображаться как **Windows.IoT**.

6. Отключите отладку и вернитесь в Visual Studio.

Упражнение 3: Программа "Hello World" в среде Blend

Blend для Visual Studio 2015 был перепроектирован, чтобы облегчить проектирование и создание приложений. В дополнение к пользовательскому интерфейсу и улучшениям производственного процесса, теперь поддерживаются XAML IntelliSense и базовые инструменты для отладки приложений. В этом упражнении мы будем использовать Blend, чтобы создать новый проект и заполнить её выборочными данными.

Задача 1 – Создаем новый проект в Blend

Blend даёт возможность начать создание нового приложения без открытия Visual Studio.

1. Запустите Blend для Visual Studio 2015. Используйте панель **Start Page (Стартовая страница)** или меню **File (Файл) > New (Новый) > Project (Проект)**, чтобы открыть диалоговое окно **New Project (Новый Проект)**.

2. **В меню Templates (Шаблоны) -> Visual C# (Визуальный C#), выберите шаблон** Blank App (Пустое приложение)**.**

3. Присвойте своему проекту имя **HelloBlend** и сохраните его в папку, в которой вы сохраняете свои проекты практикума

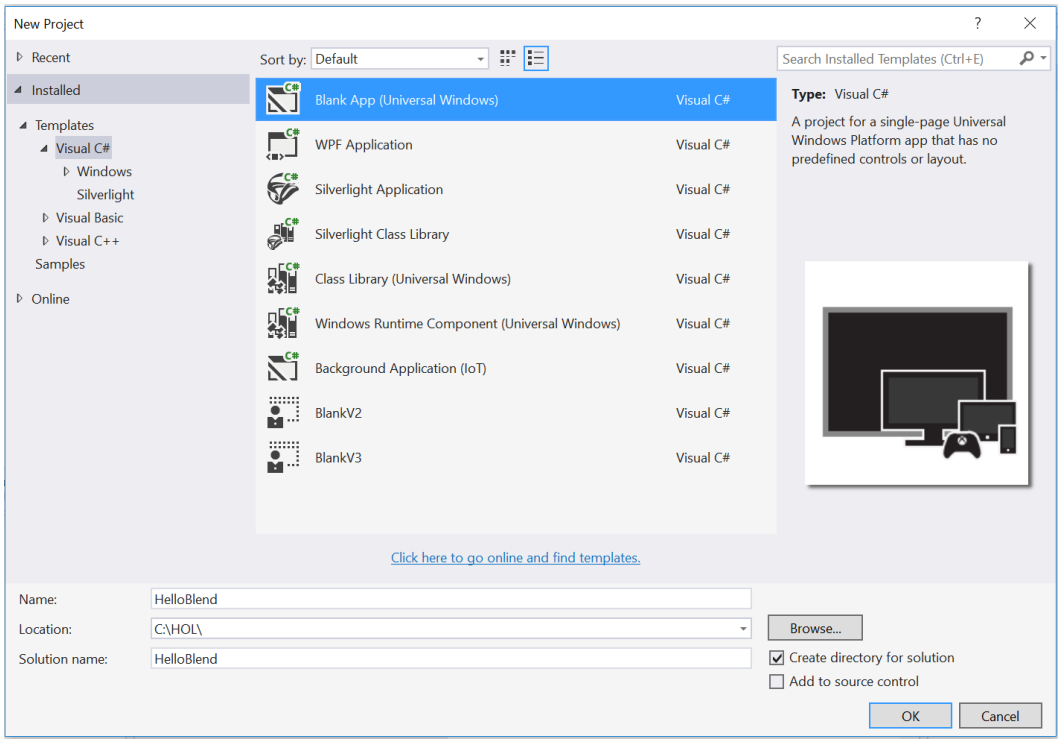

Рисунок 15

Создание проекта HelloBlend в Blend.

Примечание: Хотя вы создали этот проект в Blend, он является полностью действующим проектом под Visual Studio.

4. Blend отображает дизайн для MainPage.xaml в вашем новом проекте. Шаблон пустого приложения обеспечивает вам пустой холст для начала работы.

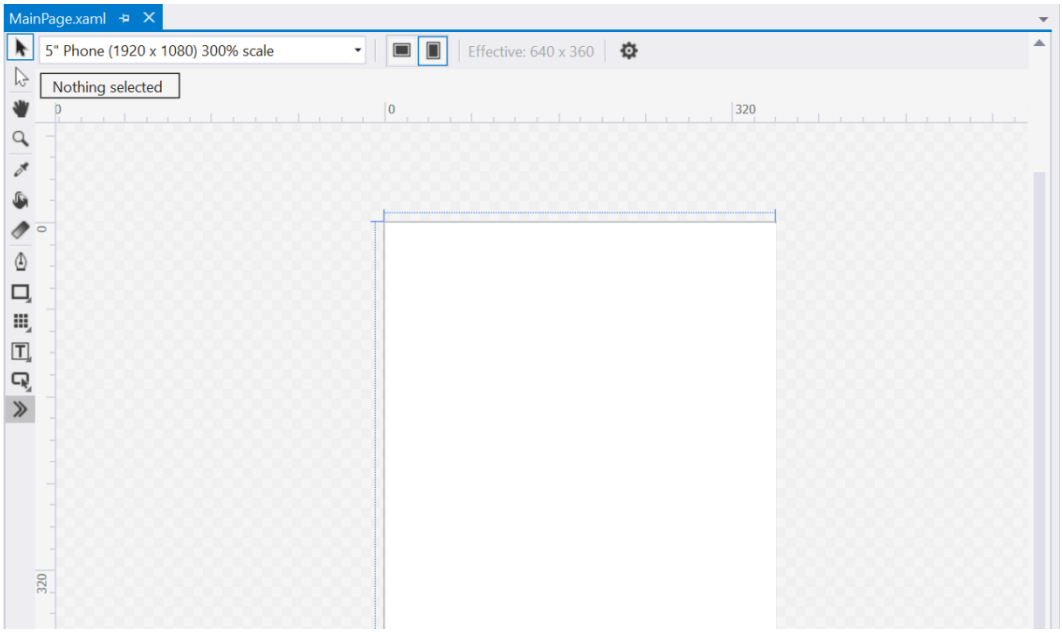

Рисунок 16

MainPage проекта пустого шаблона HelloBlend.

Задача 2 – Генерируем данные выборки

Данные выборки могут помочь начать разработку вашего приложения, ориентированного на работу с данными. Вы можете быстро сгенерировать выборочные данные в Blend через панель **Data (Данные)**. В этой задаче вы сгенерируете выборочные данные и будете просматривать их в запущенном приложении.

1. В открытом в Blend проекте **HelloBlend** перейдите в панель **Data (Данные)**. В макете окна по умолчанию панель Data (Данные) совместно использует часть интерфейса с Solution Explorer.

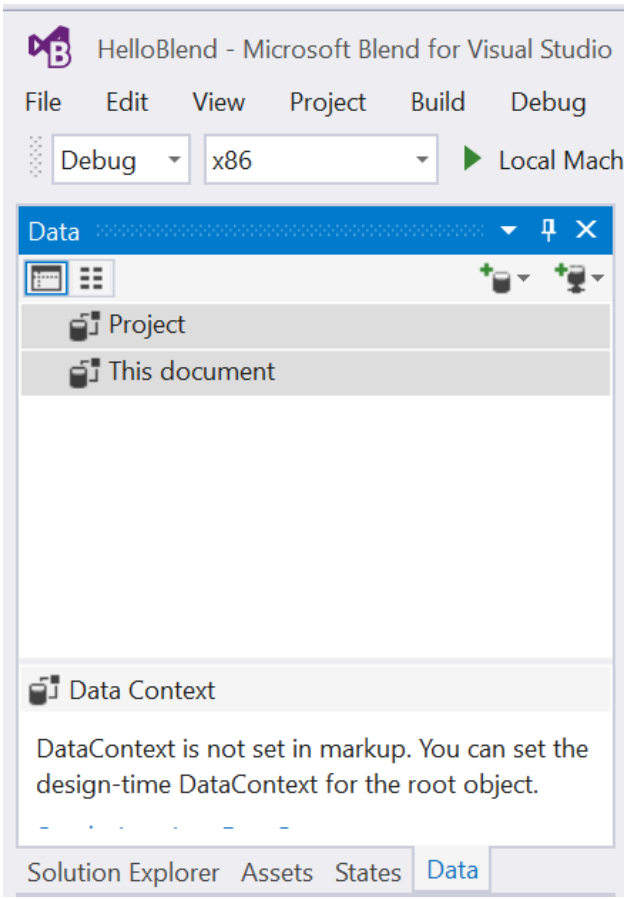

Рисунок 17

Панель данных в Blend.

Примечание: Если вы потеряете или закроете панель, вы можете набрать её название в поиске Quick Launch (Быстрый запуск).

2. В Панели данных откройте меню **Create sample data (Создать данные выборки)** и выберите **New Sample Data (Новые выборочные данные)**.

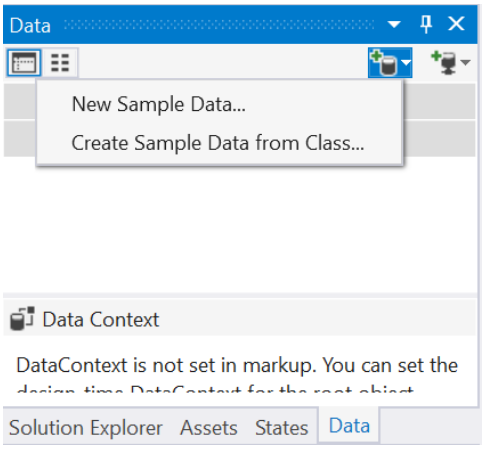

Рисунок 18

Создание выборочных данные из панели данных.

Примечание: Этот инструмент может сгенерировать данные определенного вида, а также позволяет вам использовать функцию **Create Sample Data from Class (Создать выборочные данные из класса)** для проектов, в которых вы уже определили схему данных.

3. Когда диалоговое окно **New Sample Data (Новые выборочные данные)** откроется, присвойте своему источнику данных имя и выберите, определить ли их для проекта или для текущего документа. Если вы выберете **Project (Проект)**, выборочные данные будут доступными по всем документам в проекте. Для этой демо-версии приложения мы сделаем выборочные данные доступными для текущего документа XAML, который используется в качестве MainPage (Главная страница).

4. Используйте чекбокс, чтобы включить выборку данных, когда приложение будет выполняться. Нажмите ОК для генерации выборочных данных.

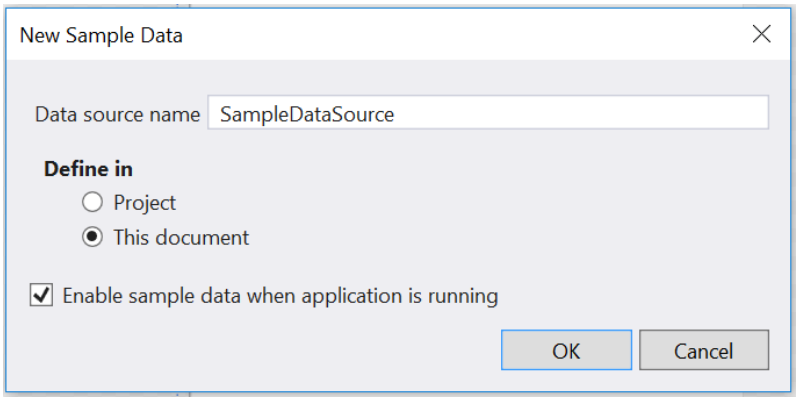

Рисунок 19
Определение выборочных данных для текущего документа XAML.

5. Ваша схема выборочных данных появится в панели **Data (Данные)** под пунктом **This document (Этот документ)**. Возможна небольшая задержка, связанна с загрузкой схемы. Когда она появится, кликните и перетащитесь на узел **Items (Элементы)**, чтобы навести курсор мыши на MainPage. Всплывающая подсказка появится, чтобы предварительно представить элементы управления, которые отобразятся в вашем приложении, если вы перетащите узел на окно приложения.

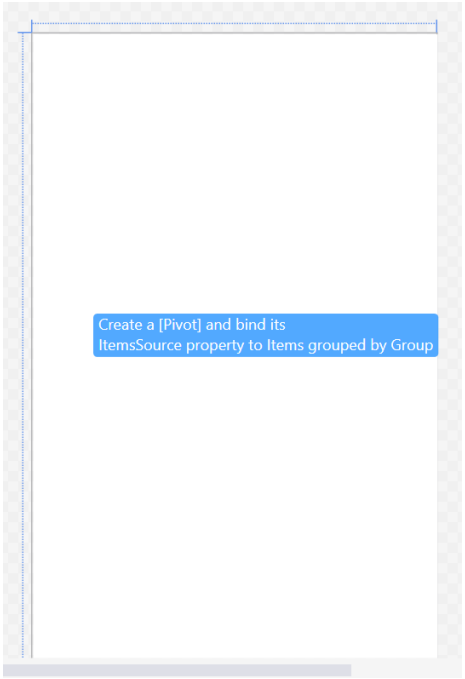

Рисунок 20

Всплывающие подсказки помогают вам предварительно просматривать выборочные данные до того, как вы добавляете их в ваше представление XAML.

6. Перетащите узел **Items (Элементы)** на страницу **MainPage**. Вы увидите образец пользовательского интерфейса для выборки данных.

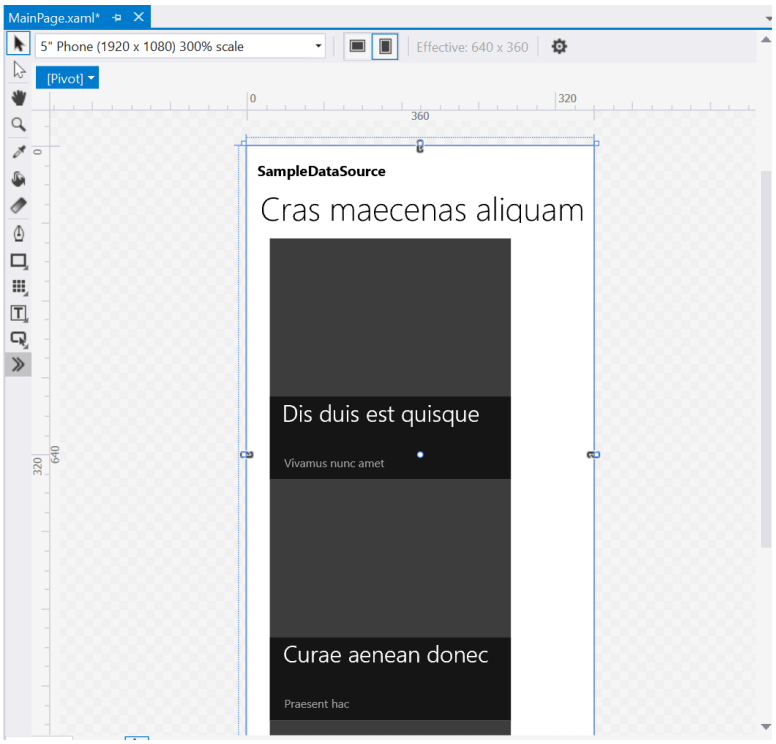

Рисунок 21

Пользовательский интерфейс для выборочных данных в средстве проектирования.

7. Соберите и запустите своё приложение на **локальном компьютере**. Используйте те же параметры отладки, которые бы вы использовали в Visual Studio. Ваши выборочные данные будут отображаться динамически в приложении.

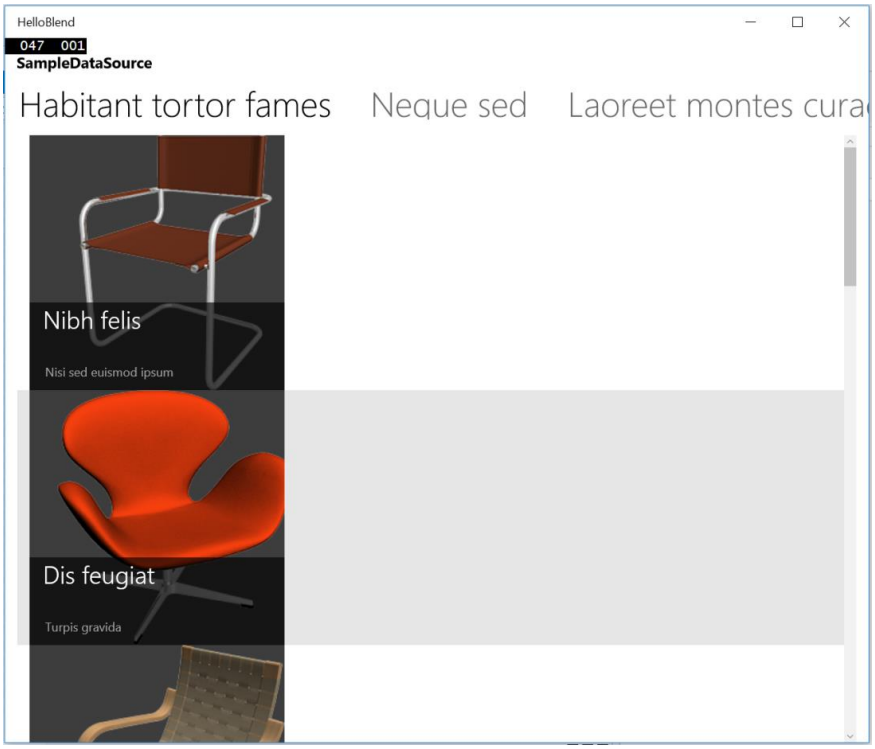

Рисунок 22
Выборочные данные в выполняющемся приложении.

8. Отключите отладку и вернитесь в Visual Studio.

9. Дополнительно: Создайте и запустите своё приложение в эмуляторе мобильной платформе или на устройстве IoT, чтобы посмотреть, как приложение будет отображаться в семействах других устройств.

Заключение

Универсальная платформа Windows является мощной коллекцией API, которая позволяет вам ориентироваться на широкий ряд устройств с помощью единственного приложения. В этом курсе вы разработали пустое приложение, созданное из шаблона в приложении Hello World, которое отображает специфическую для устройства информацию на всех устройствах под Windows 10. Вы также научились использовать выборочные данные в Blend, чтобы быстро начать создавать и визуализировать интерфейс своего приложения. В следующей работе вы научитесь, как использовать навигацию внутри приложения UWP, работать с обратной навигацией при помощи кнопки "Назад" и внедрять пользовательские элементы управления "Вперед" и "Назад".