Windows 10

Лабораторный практикум

Встраиваемая реклама

Октябрь 2015 года

Microsoft

Обзор

Пакет Universal Ad SDK прост в интегрировании, а также позволяет продвигать и монетизировать ваше приложение на различных рынках по всему миру.

В данном курсе вы научитесь устанавливать пакет Windows 10 Ad Mediator, который включает Microsoft Advertising SDK для XAML. Библиотеки Microsoft Advertising libraries для XAML/JavaScript представляют собой различные расширения из элемента управления AdMediator (Microsoft Advertising Universal SDK версии 1.0 в Справочном менеджере Visual Studio). Для ознакомления с дополнительной информацией посетите страницу: https://msdn.microsoft.com/en-us/library/mt313199(v=msads.30).aspx.

Вы будете использовать Advertising SDK для реализации встроенной рекламы на основе демо-версий, предоставляемых Microsoft.

Цели

Настоящий курс научит вас:

- Устанавливать Windows 10 Ad Mediator
- Добавлять рекламные вставки в приложение
- Добавлять рекламый банер в контент приложения

Системные требования

Для завершения настоящего курса необходимы:

- Microsoft Windows 10
- Microsoft Visual Studio 2015

Настройка

Вам следует выполнить следующие действия для подготовки компьютера:

1. Установить Microsoft Windows 10.
2. Установить Microsoft Visual Studio 2015.
3. Установить Windows Ad Mediator.

Инструкция по установке Windows Ad Mediator представлена в Упражнении 1: Задача 2.

Расчетное время для завершения курса: **15-30 минут**.

Упражнение 1: Реализация рекламы в приложении

В рамках данного упражнения вы установите пакет Windows 10 Advertising SDK и будете использовать для добавления встроенной рекламы в свое приложение.

Задача 1 – Создать пустое приложение Universal Windows

Мы начнём с создания проекта на основе шаблона Blank App (Пустого приложения).

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее **Installed (Установленное) > Templates (Шаблоны) > Visual C# > Windows > Universal**, а затем выберите шаблон **Blank App приложения (Universal Windows)**.
2. Назовите свой проект **"Advertising"** и выберите местоположение, в которое будет осуществлено сохранение результатов прохождения Лабораторного практикума. На диске **C** создана папка под именем **"HOL"**, информация о которой будет представлена в скриншотах.
3. Не изменяйте настройки, установленные для **Create new solution (Создания нового решения)** и **Create directory for solution (Создания папки для решения)**. Вы можете снять галочки как с **Add to source control (Добавить в систему контроля версий)**, так и **Show telemetry in the Windows Dev Center (Отобразить телеметрию в Windows Dev Center)**, если не хотите обновлять версию своей работы или использовать инструмент Application Insights. Нажмите **OK** для создания проекта.

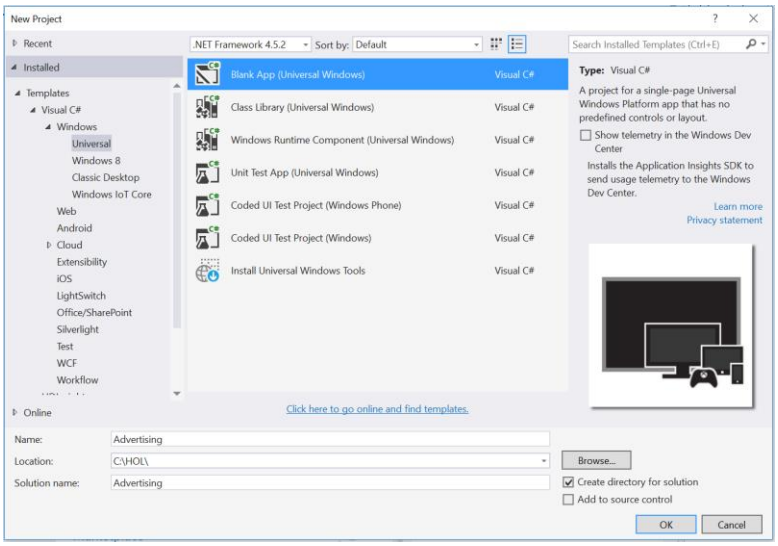

Рисунок 1

Создайте в Visual Studio 2015 новый проект Blank App.

4. Настройте Solution Configuration (Текущую конфигурацию решения) на **Debug (Отладку)** и Solution Platform (Платформу решений) в соответствии с **x86**. Выберите **Local Machine (Локальный компьютер)** из выпадающего меню Debug Target (Цели отладки).

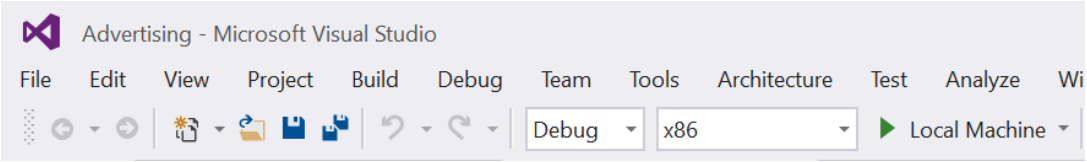

Рисунок 2

Сконфигурируйте свое приложение таким образом, чтобы оно запускалось на Local Machine (Локальном компьютере).

5. Создайте и запустите свое приложение. Вы увидите окно Blank App со счетчиком частоты кадров, активированном по умолчанию для отладки.

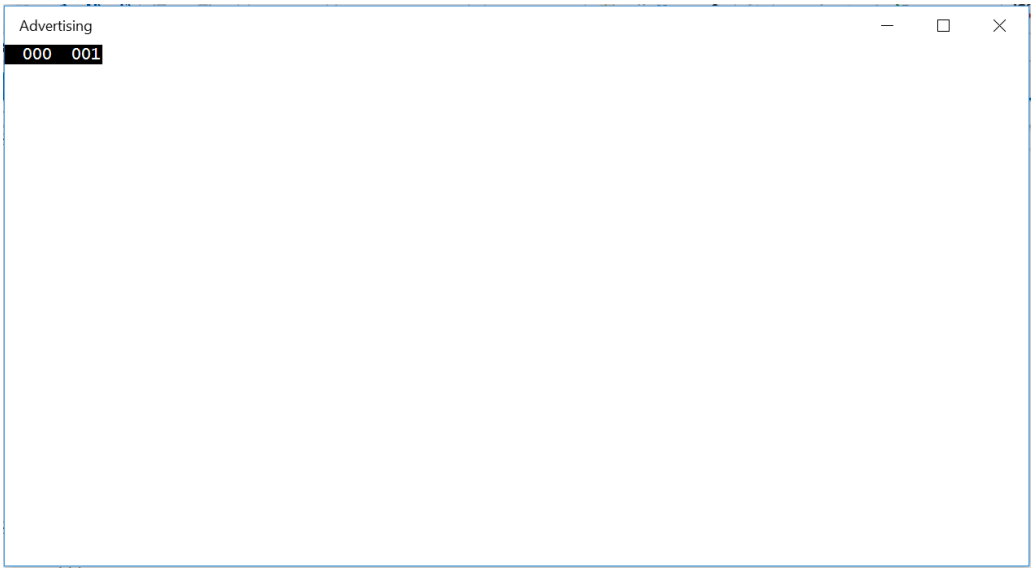

Рисунок 3

Приложение Blank universal выполняется в режиме Рабочего стола.

Примечание: Счетчик частоты кадров является инструментом, используемым в процессе отладки, который помогает следить за производительностью вашего приложения. Он полезен для тех приложений, которые требуют интенсивной графической обработки, однако не подходит для простых приложений, которые будут создаваться вами на данный момент.

В шаблоне Blank App директива препроцессора активирует или отключает счетчик частоты кадров внутри **App.xaml.cs**. Счетчик частоты кадров может перекрывать или скрывать контент вашего приложения, если не свернуть его. При выполнении данных работ вы можете отключить его, отметив **this.DebugSettings.EnableFrameRateCounter** как **False (Ложное)**.

Вернитесь в Visual Studio и остановите отладку.

Задача 2 – Установите пакет SDK для рекламы в Windows 10

Перед добавлением рекламы в ваше приложение, вам необходимо установить Windows Ad Mediator.

Примечание: Windows Ad Mediator включает пакет Microsoft Advertising SDK для XAML.

1. В Visual Studio откройте диалоговое окно Tools (Инструменты) > Extensions and Updates (Расширения и обновления).
2. Перейдите в раздел Online (Онлайн) в меню и используйте окно поиска для нахождения Ad Mediation.

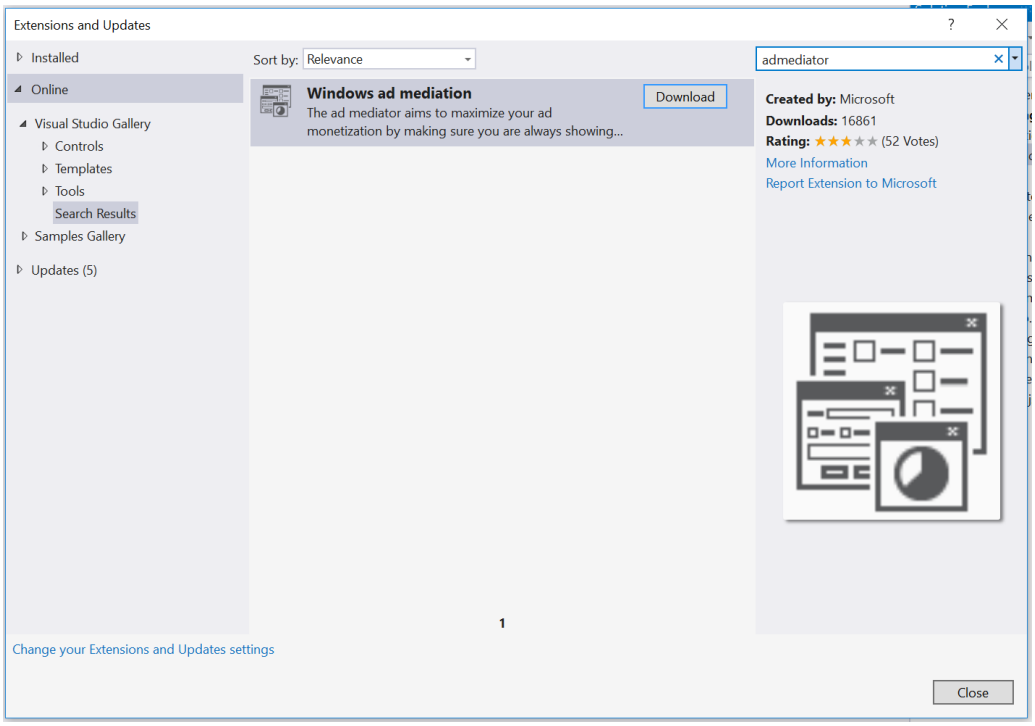

Рисунок 4

Расширение Windows Ad Mediation находится в диалоговом окне Extensions and Updates (Расширения и обновления).

3. Выберите расширение **Windows Ad Mediation**, а затем **Download (Загрузить)**. Ваш браузер начнет загрузку установочного файла.
4. Запустите установочный файл **msi**. При запуске настроек Windows Ad Mediator установите его, используя опции по умолчанию. Если всплывет окно системы User Account Control (UAC) выберите **Yes (Да)**, чтобы разрешить приложению установить программное обеспечение на ваш ПК. Когда установка будет завершена, используйте кнопку **Finish (Завершить)**, чтобы выйти из мастера настройки.

Рисунок 5

Мастер настройки Windows Ad Mediator

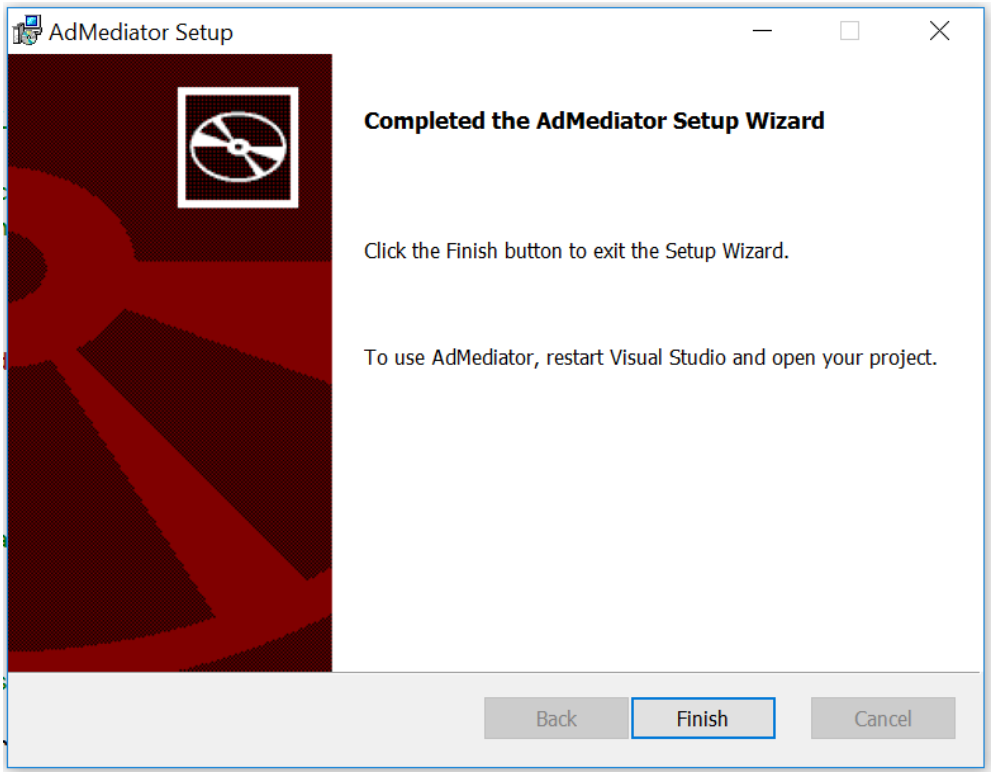

Рисунок 6

Завершите установку Windows Ad Mediator.

5. Перезапустите Visual Studio, а также откройте проект Advertising, который вы создали при выполнении Задачи 1. Когда проект будет открыт, щелкните правой кнопкой мыши по References (Ссылки) в Solutions Explorer (Обозревателе решений) и выберите **Add Reference (Добавить ссылку)**.

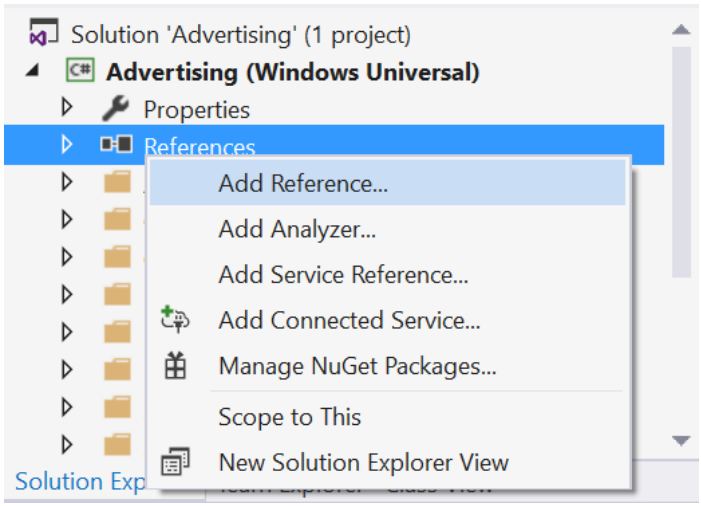

Рисунок 7

Перейдите в диалоговое окно Add Reference.

6. Разверните раздел Universal Windows и выберите **Extensions (Расширения)**. Вы увидите список пакетов SDK, применимых к вашему проекту. Отметьте галочкой **Microsoft Advertising SDK для XAML**, чтобы выбрать данный пакет и нажмите OK, чтобы добавить его в проект в качестве ссылки. Внимательно выбирайте корректный SDK!

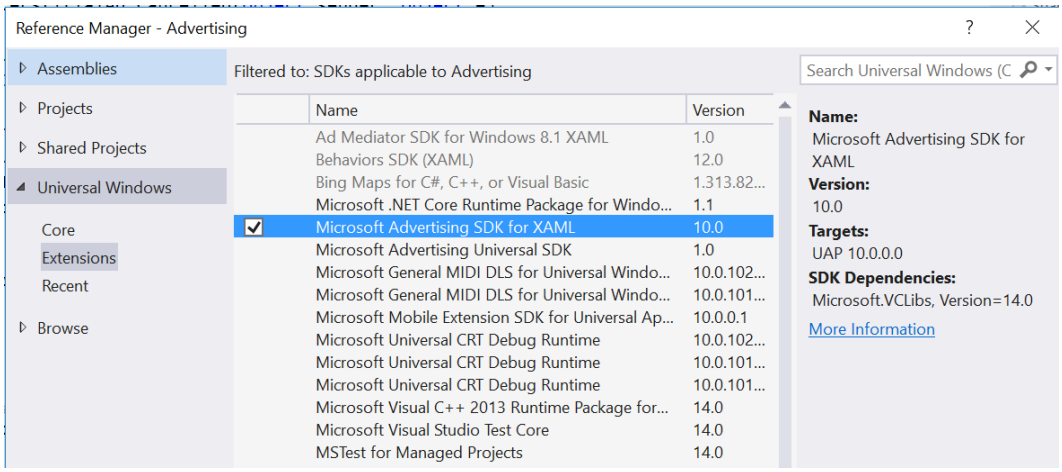

Рисунок 8

Добавьте пакет Microsoft Advertising SDK для XAML в качестве ссылки проекта.

7. После закрытия диалогового окна Add Reference, вы сможете увидеть, что Microsoft Advertising SDK появился в списке ссылок проекта.

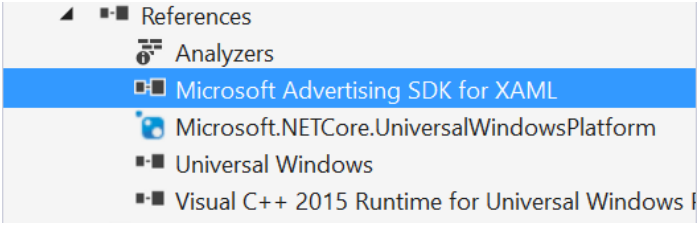

Рисунок 9

Добавьте пакет Microsoft Advertising SDK для XAML в качестве ссылки проекта.

Задача 3 – Добавление рекламной вставки в приложение

После добавления Microsoft Advertising SDK в виде ссылки к проекту Advertising, вы можете приступить к интеграции рекламы в свое приложение. Создайте новый класс, называемый DemoAds, который будет использовать тестовые файлы AppIds и AdUnits, предоставляемые Microsoft, для отображения рекламной вставки в вашем приложении.

1. Щелкните правой кнопкой мыши на наименование проекта в Solution Explorer (Обозревателе решений), а затем **Add (Добавить) > New Folder (Новая папка)**. Назовите папку **"Models"**.

2. Чтобы создать новый класс **DemoAds**, щелкните правой кнопкой мыши на папку **Models (Модели)** и выберите **Add (Добавить) > New Item (Новый элемент)**. При появлении диалогового окна **Add New Item (Добавить новый элемент)** выберите класс Visual C# в качестве нового элемента (Рисунок 17). Назовите его **DemoAds.cs**.

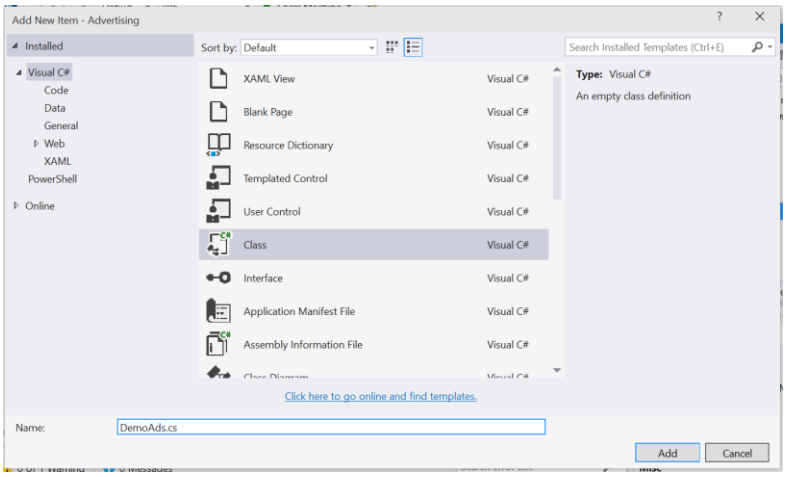

Рисунок 10

Создайте новый класс Visual C#, называемый "DemoAds".

3. Откройте **DemoAds.cs**. На данном этапе вы замените определение пустых классов рабочими классами: DemoAds и AdUnit. Код ниже отражает начальный шаблон класса, предоставляемый Visual Studio.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Advertising.Models
{
   class DemoAds
   {
   }
}
```

4. Добавьте код, выделенный красным, в файл DemoAds.cs и сохраните.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Advertising.Models
{


   /*
       These demo ad values are drawn from: https://msdn.microsoft.com/en-
US/library/mt125365(v=msads.100).aspx
   */
   public static class DemoAds
   {
       public static Dictionary<string, AdUnit> ImageAdUnits { get; private
set; }
       public static AdUnit VideoAdUnit { get; private set; }

       static DemoAds()
       {
           ImageAdUnits = new Dictionary<string, AdUnit>();

           ImageAdUnits.Add("300 x 50",
               new AdUnit { Size = "300 x 50", AdUnitId = "10865275", AppId
= "3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
           ImageAdUnits.Add("320 x 50",
               new AdUnit { Size = "320 x 50", AdUnitId = "10865270", AppId
= "3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
           ImageAdUnits.Add("300 x 250",
               new AdUnit { Size = "300 x 250", AdUnitId = "10043121",
AppId = "d25517cb-12d4-4699-8bdc-52040c712cab" });
           ImageAdUnits.Add("300 x 600",
```

```
                new AdUnit { Size = "300 x 600", AdUnitId = "10043122",
AppId = "d25517cb-12d4-4699-8bdc-52040c712cab" });
            ImageAdUnits.Add("480 x 80",
                new AdUnit { Size = "480 x 80", AdUnitId = "10865272", AppId
= "3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
            ImageAdUnits.Add("640 x 100",
                new AdUnit { Size = "640 x 100", AdUnitId = "10865273",
AppId = "3f83fe91-d6be-434d-a0ae-7351c5a997f1" });
            ImageAdUnits.Add("728 x 90",
                new AdUnit { Size = "728 x 90", AdUnitId = "10043123", AppId
= "d25517cb-12d4-4699-8bdc-52040c712cab" });

            VideoAdUnit = new AdUnit { Size = "Video", AdUnitId =
"11389925", AppId = "d25517cb-12d4-4699-8bdc-52040c712cab" };
        }
    }

    public class AdUnit
    {
        public string Size { get; set; }
        public string AdUnitId { get; set; }
        public string AppId { get; set; }
    }
}
```

Примечание: Настоящий класс ссылается на набор демо-рекламы, созданной и предоставляемой Microsoft в целях тестирования «живой рекламы» в ваших приложениях. Настоящая реклама была взята со страницы: https://msdn.microsoft.com/en-US/library/mt125365(v=msads.100).aspx

5. Откройте **MainPage.xaml.cs**. Добавьте свойства для **ShowAds** и **ViewedFullInterstitial**. Для начала сделаем так, чтобы пользователи посмотрели рекламу целиком при запуске приложения

C#

```
public sealed partial class MainPage: Страница
{
        bool _showAds = true;
        public bool ShowAds
        {
          get { return _showAds; }
          set { _showAds = value; }
        }

        bool _viewedFullInterstitial = true;
        public bool ViewedFullInterstitial
        {
          get { return _viewedFullInterstitial; }
          set { _viewedFullInterstitial = value; }
        }
```

6. Добавьте пространства имен **Microsoft.Advertising.WinRT.UI** и **Advertising.Models** для ссылки на DemoAds и Microsoft Advertising UI.

C#

```
using Microsoft.Advertising.WinRT.UI;
using Advertising.Models;

namespace Advertising
{
```

7. После определения классов добавьте приватное поле InterstitialAd.

C#

```
namespace Advertising
{
   public sealed partial class MainPage : Страница
   {
       private InterstitialAd _interstitialAd;

       bool _showAds = true;
```

8. В конструкторе страницы привяжитесь к событиям AdReady (Реклама готова), Cancelled (Отмененное), Completed (Завершенное), and ErrorOccurred (Обнаружена ошибка) и инициализируйте рекламу.

C#

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

           if (ShowAds)
           {
               // initialize the interstitial class
               _interstitialAd = new InterstitialAd();

               // wire up all 4 events
               _interstitialAd.AdReady += interstitialAd_AdReady;
               _interstitialAd.Cancelled += interstitialAd_Cancelled;
               _interstitialAd.Completed += interstitialAd_Completed;
               _interstitialAd.ErrorOccurred +=
interstitialAd_ErrorOccurred;

               RequestAd();
           }
           else
           {
               // start normally
           }
       }
```

9. Создайте обработчики событий AdReady, Cancelled, Completed и ErrorOccurred после конструктора.

C#

```
                RequestAd();
            }
            else
            {
                // start normally
            }
        }

        private void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)
        {
            // handle errors here
        }
        private void interstitialAd_Completed(object sender, object e)
        {
            // raised when the user has watched the full video
        }
        private void interstitialAd_Cancelled(object sender, object e)
        {
            // raised if the user interrupts the video
        }
        private void interstitialAd_AdReady(object sender, object e)
        {
            // raised when an ad is ready to show
        }
```

10. Добавьте код в метод **RequestAd** под конструктором. Метод будет запрашивать видео рекламу из набора демо данных, который мы определили в классе **DemoAds**.

C#

```
                RequestAd();
            }
            else
            {
                // start normally
            }
        }

        private void RequestAd()
        {
            _interstitialAd.RequestAd(AdType.Video,
DemoAds.VideoAdUnit.AppId, DemoAds.VideoAdUnit.AdUnitId);
        }

        private void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)  {
```

Примечание: В целях демонстрации мы покажем рекламную вставку, как только реклама будет готова, посредством ее добавления в обработчик событий AdReady.

11. Добавьте обработчик для события **ErrorOccurred**. Не забудьте добавить **async** в обработчик событий для ожидания завершения await.

C#

```
        private async void interstitialAd_ErrorOccurred(object sender,
AdErrorEventArgs e)
        {
            // handle errors here
            var dialog = new ContentDialog
            {
                Title = "An Error",
                Content = e.ErrorMessage,
                PrimaryButtonText = "OK",
                IsPrimaryButtonEnabled = true
            };

            await dialog.ShowAsync();
        }
```

Примечание: В настоящих приложениях может потребоваться более сложный обработчик событий.

12. Добавьте обработчик для события AdReady.

C#

```
       private void interstitialAd_AdReady(object sender, object e)
       {
           //raised when an ad is ready to show

           if (_interstitialAd.State == InterstitialAdState.Ready)
           {
               _interstitialAd.Show();
           }
       }
```

13. Создайте и запустите приложение. Рекламная видео вставка будет воспроизводиться при загрузке приложения.

Задача 4 – Запросите рекламу

Давайте сделаем так, чтобы пользователь просмотрел рекламу полностью до продолжения пользования приложением.

1.Мы используем Cancelled event посредством отображения сообщения, направленного пользователю, и перезапуска рекламы.

C#

```
private void interstitialAd_Completed(object sender, object e)
{
    // raised when the user has watched the full video
    ViewedFullInterstitial = true;

}

private async void interstitialAd_Cancelled(object sender, object e)
{
    // raised if the user interrupts the video
    var dialog = new ContentDialog
    {
        Title = "Ad Interrupted",
        Content = "You must watch the complete ad!",
        PrimaryButtonText = "OK",
        IsPrimaryButtonEnabled = true
    };

    await dialog.ShowAsync();

    RequestAd();
}
```

Примечание: Поскольку мы собираемся использовать элемент управления ContentDialog для отображения сообщения об ошибке посредством асинхронного вызова, необходимо добавить async в определение метода обработчика событий interstitialAd_Cancelled. ContentDialog является новым элементом управления в Windows 10, который облегчает отображение разнообразного контента через модальное диалоговое окно приложения. Для ознакомления с дополнительной информацией о ContentDialogs посетите страницу: https://msdn.microsoft.com/library/windows/apps/windows.ui.xaml.controls.contentdialog.aspx

2.Постройте и запустите приложение. Щелкните на запущенное видео для отображения кнопки Back (Назад). Нажмите кнопку Back во время просмотра рекламы чтобы посмотреть почедение метода interstitialAd_Cancelled.
3.Отключите отладку и вернитесь в Visual Studio.

Было бы неудобно не иметь возможности пропустить рекламу во время выполнения данных упражнений. Чтобы предоставить возможность пропуска рекламы, закомментируйте обработчик события.

C#

```
private async void interstitialAd_Cancelled(object sender, object e)
{
    // raised if the user interrupts the video
    //var dialog = new ContentDialog
    //{
    // Title = "Ad Interrupted",
    // Content = "You must watch the complete ad!",
```

```
            // PrimaryButtonText = "OK",
            // IsPrimaryButtonEnabled = true
            //};

            //await dialog.ShowAsync();

            //RequestAd();
        }
```

4.Создайте и повторно запустите свое приложение. Убедитесь, что вы можете пропустить рекламу, выбрав кнопку Back во время ее воспроизведения.
5.Отключите отладку и вернитесь в Visual Studio.

Задача 5 – Отобразите рекламный банер

В рамках этой задачи вы будете использовать Microsoft Advertising SDK для отображения встроенной рекламы сбоку от контента вашего приложения. Схожие приемы могут быть использованы, чтобы разместить рекламу, встроенную в другие элементы управления.

1. Откройте **MainPage.xaml**. Добавьте пространство имен Microsoft.Advertising.WinRT.UI**.**

XAML

```
    xmlns:UI="using:Microsoft.Advertising.WinRT.UI"
```

2.Чтобы создать контейнер для вашей встроенной рекламы, замените элементы **Grid** в **MainPage.xaml** на **Hub**. Добавьте hub-раздел **Advertising (Реклама)** и соседний раздел **Content (Контент)** в концентратор (hub). Hub-раздел Advertising использует **AdControl** из Advertising SDK, чтобы отобразить статичный рекламный банер слева от контента приложения. В окне дизайнера интерфейса, данный hub-раздел отобразит синий прямоугольник с теми же размерами, что и сама реклама. Установите параметр Visibility в значение Collapsed.

XAML

```
<Hub Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <HubSection
            VerticalContentAlignment="Stretch"
            x:Name="AdvertisingSection"
            Header="Advertising"
            Visibility="Collapsed">
            <DataTemplate>
                <Grid VerticalAlignment="Top">
                    <!-- The rectangle acts as a place holder so we can see
where the ad control is located-->
                    <Rectangle Fill="Blue" Width="300" Height="600"/>
                    <UI:AdControl
                        ApplicationId="d25517cb-12d4-4699-8bdc-
52040c712cab"
                        AdUnitId="10043122"
```

```
                        Height="600"
                        VerticalAlignment="Top"
                        Width="300"/>
                </Grid>
            </DataTemplate>
        </HubSection>
        <HubSection Header="Content" />
</Hub>
```

3.Во вспомогательном коде MainPage добавьте метод ShowInlineAds() для отображения AdvertisingSection, если **ShowAds** – True.

C#

```
private void ShowInlineAds()
{
    if (ShowAds)
    {
        AdvertisingSection.Visibility = Visibility.Visible;
    }
    else
    {
        AdvertisingSection.Visibility = Visibility.Collapsed;
    }
}
```

4.Вызовите метод **ShowInlineAds()** после **RequestAd()** в конструкторе.

C#

```
    RequestAd();
    ShowInlineAds();
}
```

5.Постройте и запустите свое приложение. После того окончания воспроизведения рекламной вставки вы увидите, что рекламный банер появится слева от контента вашего приложения.

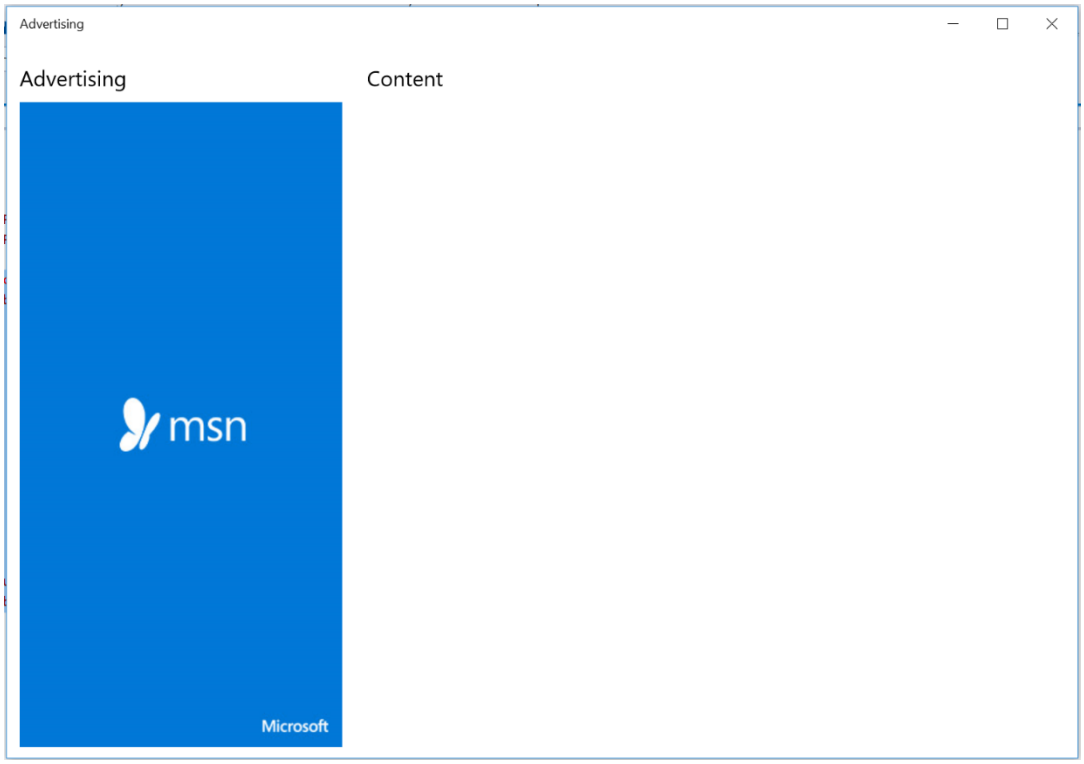

Рисунок 11

Встроенная реклама в приложении.

6. Завершите отладку и вернитесь в Visual Studio.

Краткий обзор

В рамках настоящего курса вы научились загружать и устанавливать Advertising SDK,и отображать новые рекламные видео вставки. Теперь вы можете делать так, чтобы пользователи в обязательном порядке просматривали вашу рекламу, а также добавлять рекламу на основе изображений в элемент управления Hub.