

**Лабораторный ­практикум**

Курс: Интегрируемая реклама

Сентябрь 2015 года

Содержание

**No table of contents entries found.**

Обзор

1. Пакет Universal Ad SDK прост в интегрировании, а также предоставляет вам возможность осуществлять продвижение и монетизировать ваше приложение на различных рынках по всему миру.

В рамках данного курса вы научитесь устанавливать пакет Windows 10 Ad Mediator, который включает Microsoft Advertising SDK для XAML. Библиотеки рекламы 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 рекламной вставки и встроенной рекламы на основе демо-версий, предоставляемых Microsoft.

# Цели

* 1. Настоящий курс научит вас:
  + Устанавливать Windows 10 Ad Mediator
  + Реализовывать рекламную вставку в приложении
  + Добавлять встроенную рекламу в контент приложения
  + Реализовывать ValueWhenConverter с целью переключения режима видимости рекламы

# Системные требования

* 1. Для завершения настоящего курса необходимы:
  + Microsoft Windows 10
  + Microsoft Visual Studio 2015

# Настройка

* 1. Вам следует выполнить следующие действия для подготовки компьютера:
  2. Установить Microsoft Windows 10.
  3. Установить Microsoft Visual Studio 2015.
  4. Установить установочный пакет Windows Ad Mediator.

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

# Упражнения

* 1. Настоящий Лабораторный практикум включает следующие упражнения:
  2. Реализация Рекламы в Приложении
  3. Расчетное время для завершения курса: **15-30 минут**.

Упражнение 1: Реализация рекламы в приложении

1. В рамках данного упражнения вы установите пакет Windows 10 Advertising SDK и будете использовать для добавления рекламной вставки и 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 файловой системы, в которое будет осуществлено сохранение результатов прохождения Лабораторного практикума. На диске **C** создана папка под именем **"HOL"**, информация о которой будет представлена в скриншотах.

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.

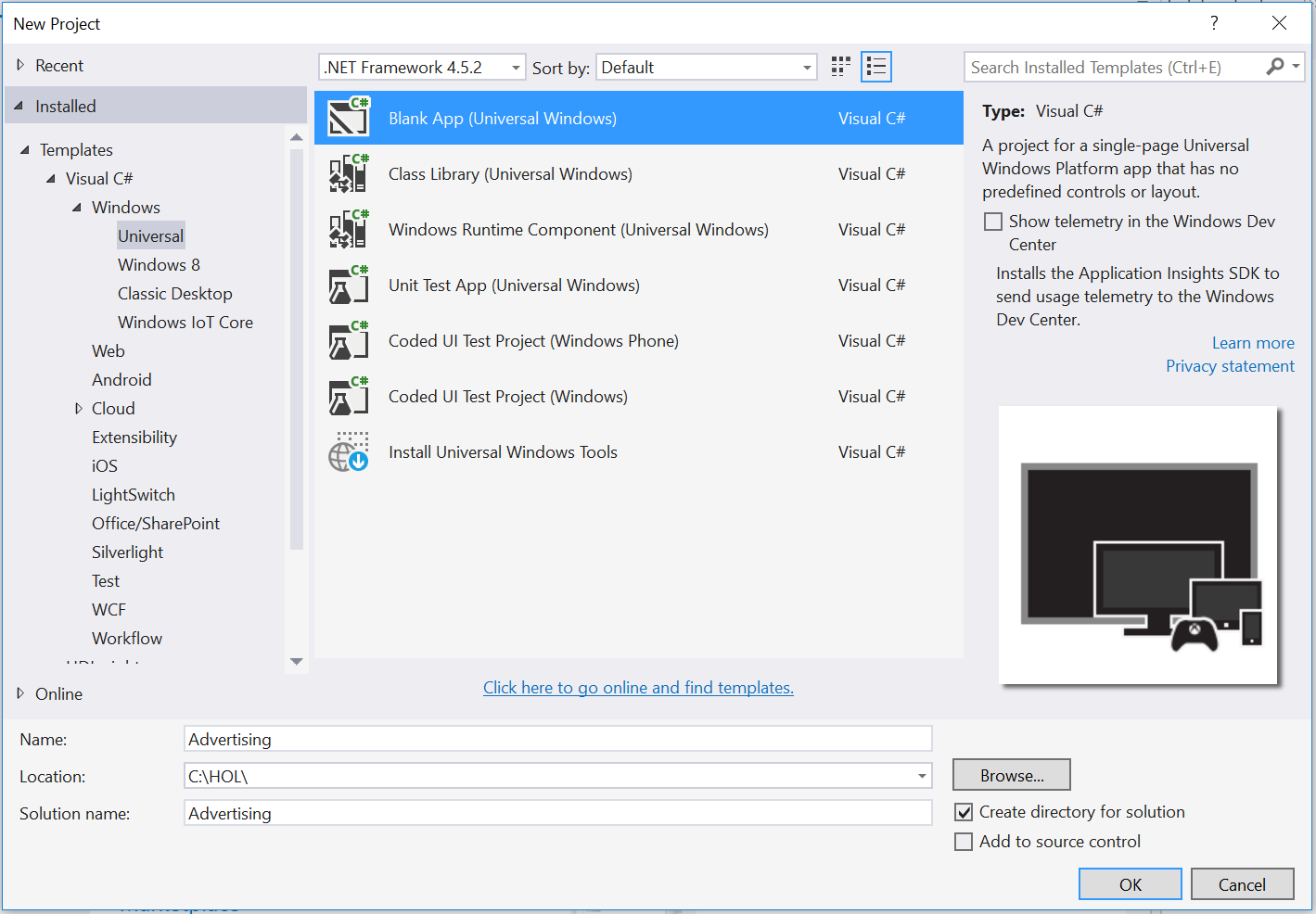


Рисунок 1

Создайте в Visual Studio 2015 новый проект Blank App.

1. 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).

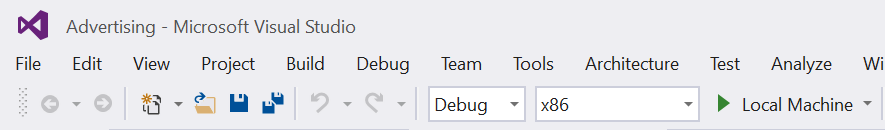


Рисунок 2

* + 1. Сконфигурируйте свое приложение таким образом, чтобы оно запускалось на Local Machine (Локальном компьютере).

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

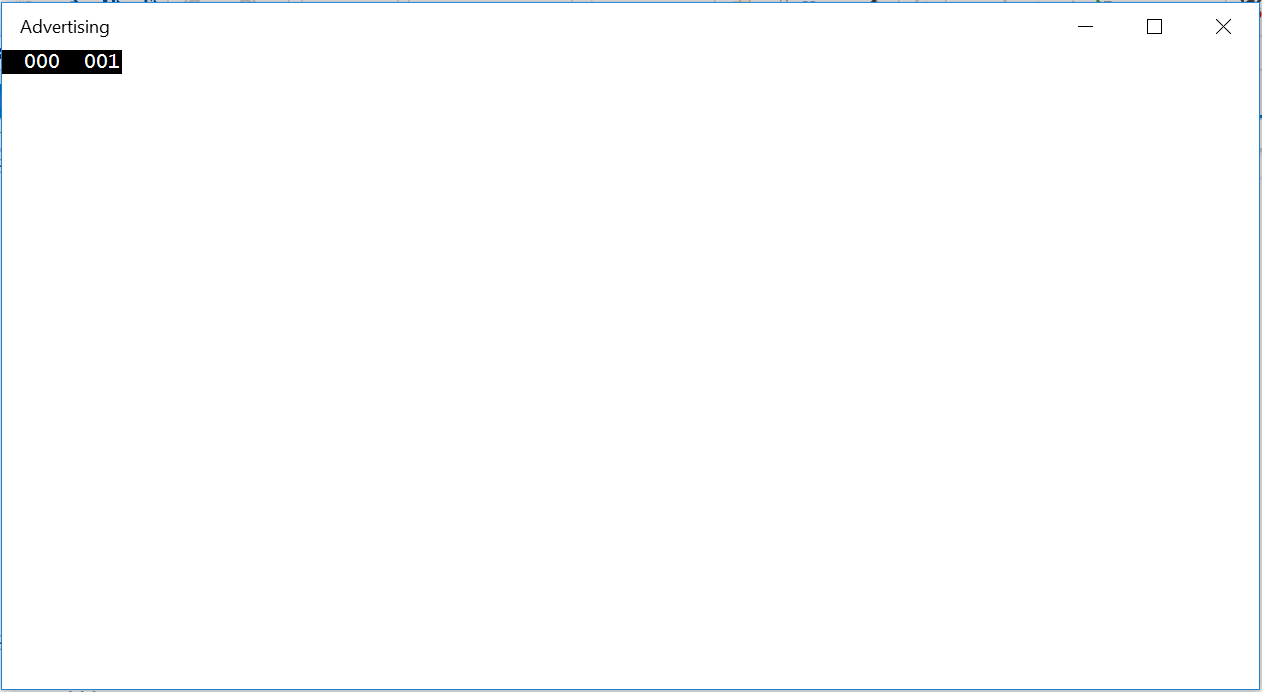


Рисунок 3

Приложение Blank universal выполняется в режиме Рабочего стола.

* 1. **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.
  2. В шаблоне Blank App директива препроцессора активирует или отключает счетчик частоты кадров посредством **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)**.

1. Вернитесь в Visual Studio и отключите отладку.

Задача 2 – Установите пакет SDK для рекламы в Windows 10

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

* 1. **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**.

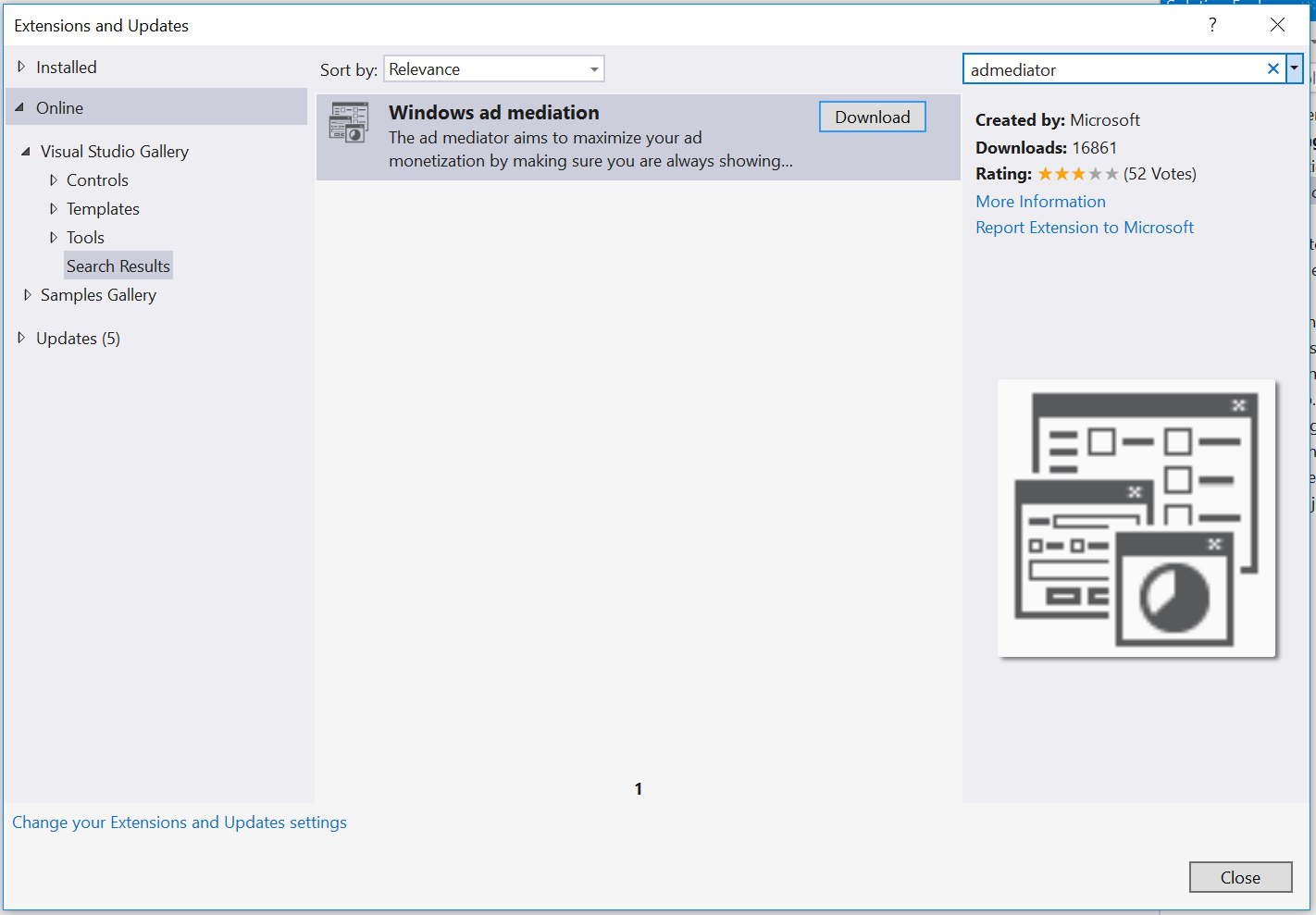


Рисунок 4

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

1. Выберите расширение **Windows Ad Mediation**, а затем **Download (Загрузить)**. Ваш браузер начнет загрузку установочного файла.
2. Запустите установочный файл **msi**. При запуске настроек Windows Ad Mediator установите его, используя опции по умолчанию. При запросе об открытии от имени Пользователя 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.

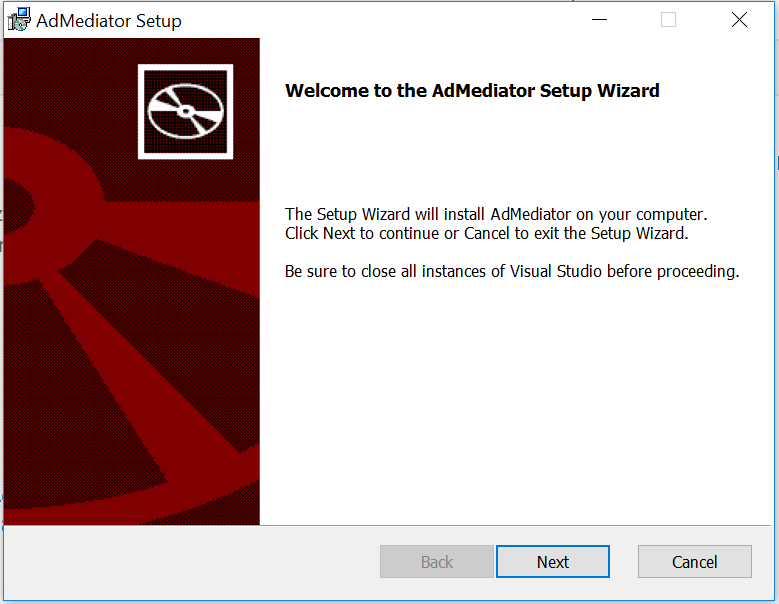


Рисунок 5

Мастер настройки Windows Ad Mediator.

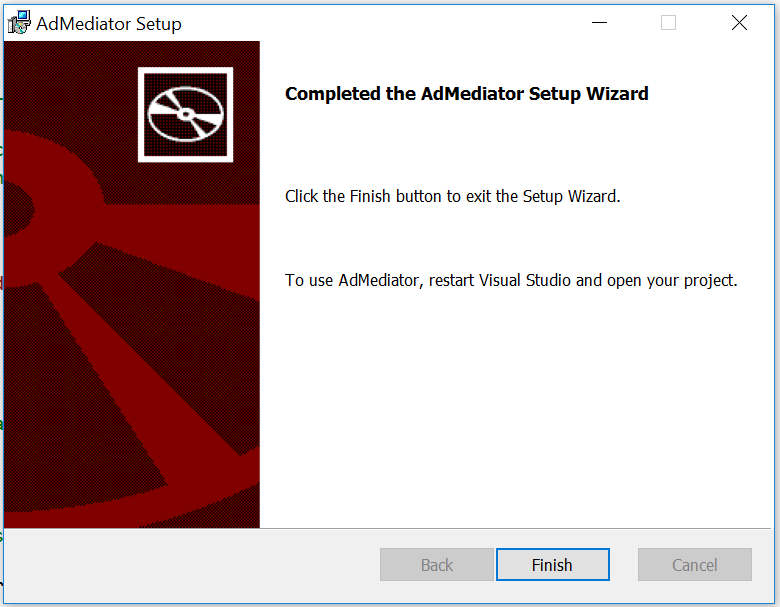


Рисунок 6

Завершите установку Windows Ad Mediator.

1. Закройте и снова откройте Visual Studio, а также загрузите 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)**.

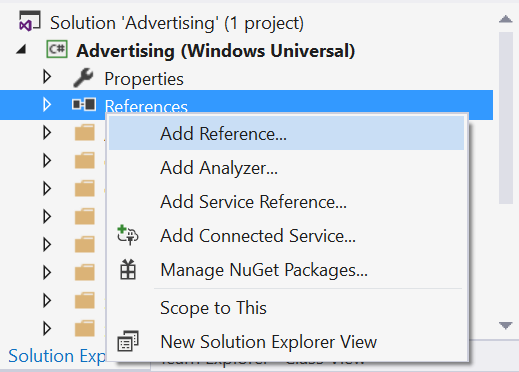


Рисунок 7

Перейдите в диалоговое окно Add Reference.

1. Разверните раздел Windows Universal 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!

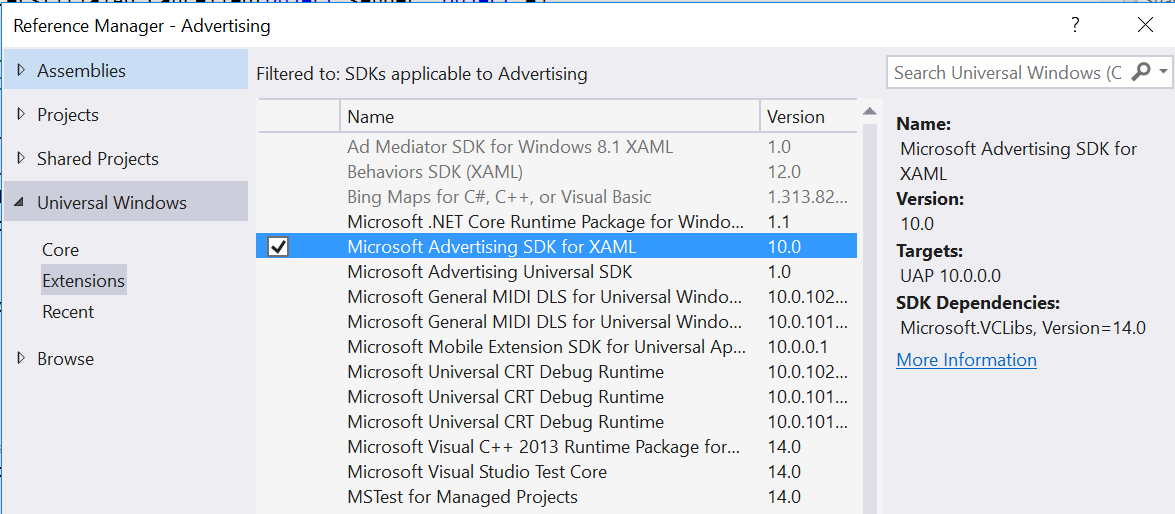


Рисунок 8

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

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

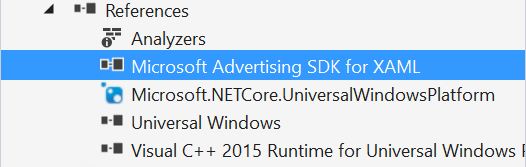


Рисунок 9

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

Задача 3 – Добавьте рекламную вставку

* 1. После представления Microsoft Advertising SDK в виде ссылки проекта Advertising, вы можете приступить к интеграции рекламы в свое приложение. В рамках настоящей задачи вы создадите 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). Н **DemoAds.cs**.

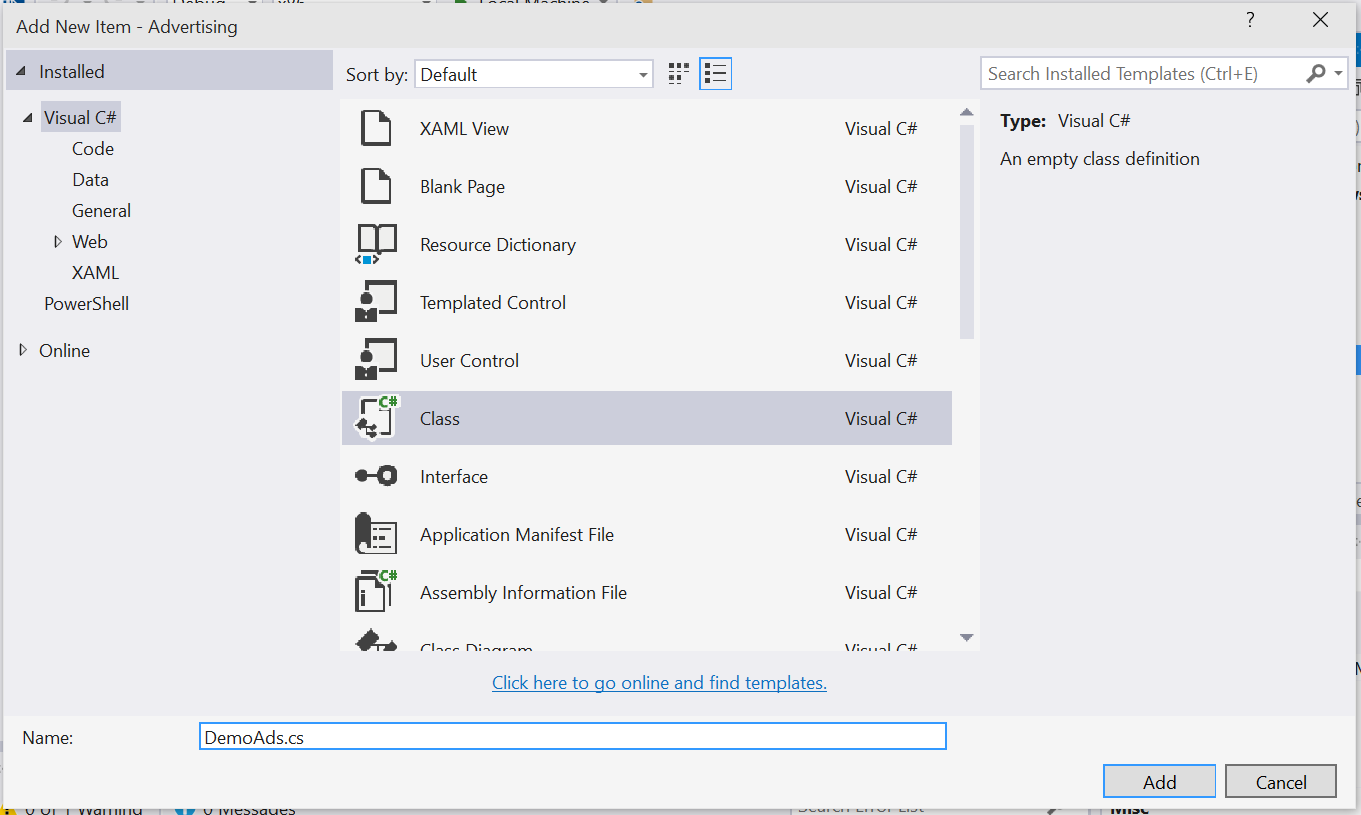


Рисунок 10

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

1. Откройте **DemoAds.cs**. На данном этапе вы замените определение пустых классов рабочими классами: DemoAds и AdUnit. Кодовый пакет ниже отражает начальный пункт, предоставляемый Visual Studio.

C#

* 1. using System;
  2. using System.Collections.Generic;
  3. using System.Linq;
  4. using System.Text;
  5. using System.Threading.Tasks;
  6. namespace Advertising.Models
  7. {

   class DemoAds

   {

   }

* 1. }

Замените определение пустых классов в пакете предыдущего кода следующим определением классов, выделеным красным. Сохраните DemoAds.cs.

C#

* 1. using System;
  2. using System.Collections.Generic;
  3. using System.Linq;
  4. using System.Text;
  5. using System.Threading.Tasks;
  6. namespace Advertising.Models
  7. {

   /\*

       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; }

   }

* 1. }
  2. **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>

1. Откройте **MainPage.xaml.cs**. Добавьте свойства для **ShowAds** и **ViewedFullInterstitial**. Вы можете сделать так, чтобы перед продолжением работы приложения, пользователи должны были полностью посмотреть рекламу.

C#

* 1. public sealed partial class MainPage: Страница
  2. {
  3. bool \_showAds = true;
  4. public bool ShowAds
  5. {
  6. get { return \_showAds; }
  7. set { \_showAds = value; }
  8. }
  9. bool \_viewedFullInterstitial = true;
  10. public bool ViewedFullInterstitial
  11. {
  12. get { return \_viewedFullInterstitial; }
  13. set { \_viewedFullInterstitial = value; }
  14. }

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

C#

* 1. using Microsoft.Advertising.WinRT.UI;
  2. using Advertising.Models;
  3. namespace Advertising
  4. {

1. После определения классов добавьте закрытое поле InterstitialAd.

C#

* 1. namespace Advertising
  2. {
  3. public sealed partial class MainPage : Страница
  4. {
  5. private InterstitialAd \_interstitialAd;
  6. bool \_showAds = true;

1. Инициализируйте класс для рекламной вставки и привяжите его к событиям в конструкторе: AdReady (Реклама готова), Cancelled (Отмененное), Completed (Завершенное), and ErrorOccurred (Обнаружена ошибка).

C#

* 1. public MainPage()
  2. {
  3. this.InitializeComponent();
  5. if (ShowAds)
  6. {
  7. // initialize the interstitial class
  8. \_interstitialAd = new InterstitialAd();
  9. // wire up all 4 events
  10. \_interstitialAd.AdReady += interstitialAd\_AdReady;
  11. \_interstitialAd.Cancelled += interstitialAd\_Cancelled;
  12. \_interstitialAd.Completed += interstitialAd\_Completed;
  13. \_interstitialAd.ErrorOccurred += interstitialAd\_ErrorOccurred;
  14. RequestAd();
  15. }
  16. else
  17. {
  18. // start normally
  19. }
  20. }

1. Создайте события AdReady, Cancelled, Completed и ErrorOccurred в определении конструктора.

C#

* 1. RequestAd();
  2. }
  3. else
  4. {
  5. // start normally
  6. }
  7. }
  8. private void interstitialAd\_ErrorOccurred(object sender, AdErrorEventArgs e)
  9. {
  10. // handle errors here
  11. }
  12. private void interstitialAd\_Completed(object sender, object e)
  13. {
  14. // raised when the user has watched the full video
  15. }
  16. private void interstitialAd\_Cancelled(object sender, object e)
  17. {
  18. // raised if the user interrupts the video
  19. }
  20. private void interstitialAd\_AdReady(object sender, object e)
  21. {
  22. // raised when an ad is ready to show
  23. }

1. Добавьте определение для события **RequestAd** под конструктором. Настоящий обработчик запрашивает видеорекламу от набора рекламы в виде демо-версий, на который вы ссылались в классе **DemoAds**.

C#

* 1. RequestAd();
  2. }
  3. else
  4. {
  5. // start normally
  6. }
  7. }
  8. private void RequestAd()
  9. {
  10. \_interstitialAd.RequestAd(AdType.Video, DemoAds.VideoAdUnit.AppId, DemoAds.VideoAdUnit.AdUnitId);
  11. }
  12. private void interstitialAd\_ErrorOccurred(object sender, AdErrorEventArgs e)
  13. {
  14. **Примечание:** В целях настоящего курса мы покажем рекламную вставку, как только реклама будет готова, посредством ее добавления в обработчик событий AdReady.

1. Добавьте определение для события **ErrorOccurred**. Убедитесь в необходимости добавления **async** в обработчик событий.
   * 1. C#

private async void interstitialAd\_ErrorOccurred(object sender, AdErrorEventArgs e)

* 1. {
  2. // handle errors here
  3. var dialog = new ContentDialog
  4. {
  5. Title = "An Error",
  6. Content = e.ErrorMessage,
  7. PrimaryButtonText = "OK",
  8. IsPrimaryButtonEnabled = true
  9. };
  10. await dialog.ShowAsync();
  11. }
  12. **Примечание:** При работе Приложения на уровне предприятия может потребоваться более устойчивая обработка ошибок.

1. Добавьте определение для события AdReady.

C#

* 1. private void interstitialAd\_AdReady(object sender, object e)
  2. {
  3. //raised when an ad is ready to show
  5. if (\_interstitialAd.State == InterstitialAdState.Ready)
  6. {
  7. \_interstitialAd.Show();
  8. }
  9. }

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

Задача 4 – Запросите рекламу

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

1. Мы используем Отмененное событие (Cancelled event) посредством отображения сообщения, направленного пользователю, и перезапуска рекламы.

C#

private void interstitialAd\_Completed(object sender, object e)

* 1. {
  2. // raised when the user has watched the full video
  3. ViewedFullInterstitial = true;
  4. }
  5. private async void interstitialAd\_Cancelled(object sender, object e)
  6. {
  7. // raised if the user interrupts the video
  8. var dialog = new ContentDialog
  9. {
  10. Title = "Ad Interrupted",
  11. Content = "You must watch the complete ad!",
  12. PrimaryButtonText = "OK",
  13. IsPrimaryButtonEnabled = true
  14. };
  15. await dialog.ShowAsync();
  16. RequestAd();
  17. }
  18. **Примечание:** Поскольку мы собираемся использовать элемент управления ContentDialog для отображения сообщения об ошибке посредством метода async, мы должны быть уверены в желании 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>

1. Создайте и запустите приложение. Щелкните на запущенное видео для отображения кнопки Back (Назад). Выберите кнопку Back во время просмотра рекламы с целью оценки поведения события interstitialAd\_Cancelled.
2. Отключите отладку и вернитесь в Visual Studio.
3. При прокрутке вперед было бы неудобно не иметь возможности пропустить рекламу во время выполнения данных упражнений. Чтобы предоставить возможность пропуска рекламы, оставьте комментарий к содержанию обработчика отмененных событий.

C#

private async void interstitialAd\_Cancelled(object sender, object e)

* 1. {
  2. // raised if the user interrupts the video
  3. //var dialog = new ContentDialog
  4. //{
  5. // Title = "Ad Interrupted",
  6. // Content = "You must watch the complete ad!",
  7. // PrimaryButtonText = "OK",
  8. // IsPrimaryButtonEnabled = true
  9. //};
  10. //await dialog.ShowAsync();
  11. //RequestAd();
  12. }

1. Создайте и повторно запустите свое приложение. Подтвердите, что вы можете пропустить рекламу, выбрав кнопку Back во время ее воспроизведения.
2. Отключите отладку и вернитесь в Visual Studio.
3. Задача 4 – Отобразите встроенную рекламу
   1. В рамках этой задачи вы будете использовать Microsoft Advertising SDK для отображения встроенной рекламы в отдельном столбце вашего приложения. Схожие приемы могут быть использованы, чтобы разместить рекламу, встроенную в другие элементы управления.
4. Откройте **MainPage.xaml**. Добавьте пространство именMicrosoft.Advertising.WinRT.UI**.**

XAML

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

1. 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 статистику рекламы слева от контента приложения. С точки зрения дизайна, данный hub-раздел отобразит синий прямоугольник с теми же размерами, что и сама реклама. Установите видимость в соответствии с настройками по умолчанию.

XAML

* 1. <Hub Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
  2. <HubSection
  3. VerticalContentAlignment="Stretch"
  4. x:Name="AdvertisingSection"
  5. Header="Advertising"
  6. Visibility="Collapsed">
  7. <DataTemplate>
  8. <Grid VerticalAlignment="Top">
  9. <!-- The rectangle acts as a place holder so we can see where the ad control is located-->
  10. <Rectangle Fill="Blue" Width="300" Height="600"/>
  11. <UI:AdControl
  12. ApplicationId="d25517cb-12d4-4699-8bdc-52040c712cab"  
       AdUnitId="10043122"
  13. Height="600"
  14. VerticalAlignment="Top"
  15. Width="300"/>
  16. </Grid>
  17. </DataTemplate>
  18. </HubSection>
  19. <HubSection Header="Content" />
  20. </Hub>

1. Во вспомогательном коде MainPage добавьте метод ShowInlineAds() для отображения AdvertisingSection, если **ShowAds** – True (истинен).
   * 1. C#
   1. private void ShowInlineAds()
   2. {
   3. if (ShowAds)
   4. {
2. AdvertisingSection.Visibility = Visibility.Visible;
3. }
4. else
5. {

AdvertisingSection.Visibility = Visibility.Collapsed;

}

* 1. }

1. Вызовите метод **ShowInlineAds()** после **RequestAd()** в конструкторе.
   * 1. C#
   1. RequestAd();
   2. ShowInlineAds();
   3. }
2. Создайте и запустите свое приложение. После того окончания воспроизведения рекламной вставки вы увидите, что изображение встроенной рекламы появится слева от контента вашего приложения.

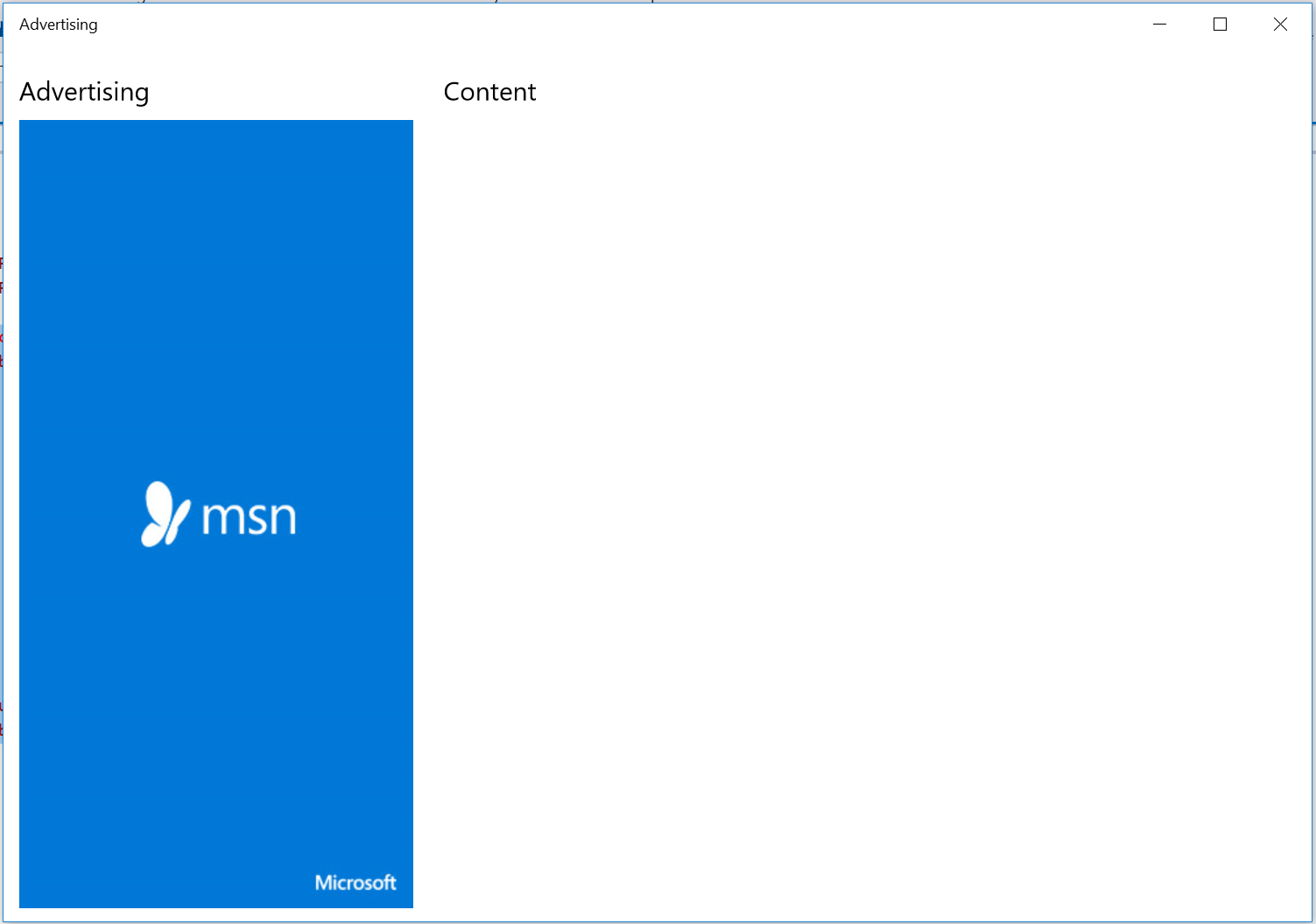


Рисунок 11

Встроенная реклама в приложении.

1. Отключите отладку и вернитесь в Visual Studio.

Краткий обзор

* 1. В рамках настоящего курса вы научились загружать и устанавливать Advertising SDK, как 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 Концентратора (Hub).