

Лабораторный­ практикум

Курс: «Живые плитки» и уведомления

Сентябрь 2015 года

Содержание

**No table of contents entries found.**

Обзор

Windows 10 предоставляет уникальные и привлекательные способы для взаимодействия с пользователями, которые выходят за рамки привычного. Теперь вы можете задать необходимый контент для отдельной «Живой плитки» посредством использования новой адаптивной разметки, которая позволит максимально эффективно использовать размер плитки и плотность экрана. Всплывающие уведомления (Toast) могут включать интерактивные и адаптивные элементы, изображения, а также способны синхронизироваться с «Живыми плитками».

# Цели

* 1. Настоящий курс научит вас:
  + Задавать активы графических изображений и цвет фона для плиток по умолчанию
  + Использовать BadgeUpdateManager для обновления счетчика уведомления плитки
  + Создавать адаптивные шаблоны для «Живых плиток»
  + Обновлять «Живые плитки»
  + Использовать маленькие, средние, широкие и большие плитки
  + Создавать всплывающие уведомления (Toast) с выбираемыми пользователем элементами
  + Управлять всплывающими уведомлениями (Toast) посредством активации фона

# Системные требования

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

# Настройка

* 1. Вы должны выполнить следующие действия для подготовки компьютера:
  2. Установить Microsoft Windows 10.
  3. Установить Microsoft Visual Studio 2015.

# Упражнения

* 1. Настоящий Лабораторный практикум включает следующие упражнения:
  2. Настройка плитки по умолчанию
  3. Запуск интерактивного всплывающего уведомления (Toast)
  4. Контролировать обновление плиток
  5. Расчетное время для завершения курса: **45-60 минут**.

Упражнение 1: Настройка плитки по умолчанию

1. Приложения Windows 10 используют выбранный вами логотип (изображение) для отображения плитки по умолчанию. В данном упражнении вы импортируете логотип (изображение) с целью создания простых плиток и обновления уведомления плитки при использовании BadgeUpdateManager.

Задача 1 – Создать шаблон приложения Universal Windows

Мы начнем 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 **"TilesAndNotifications"** 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.

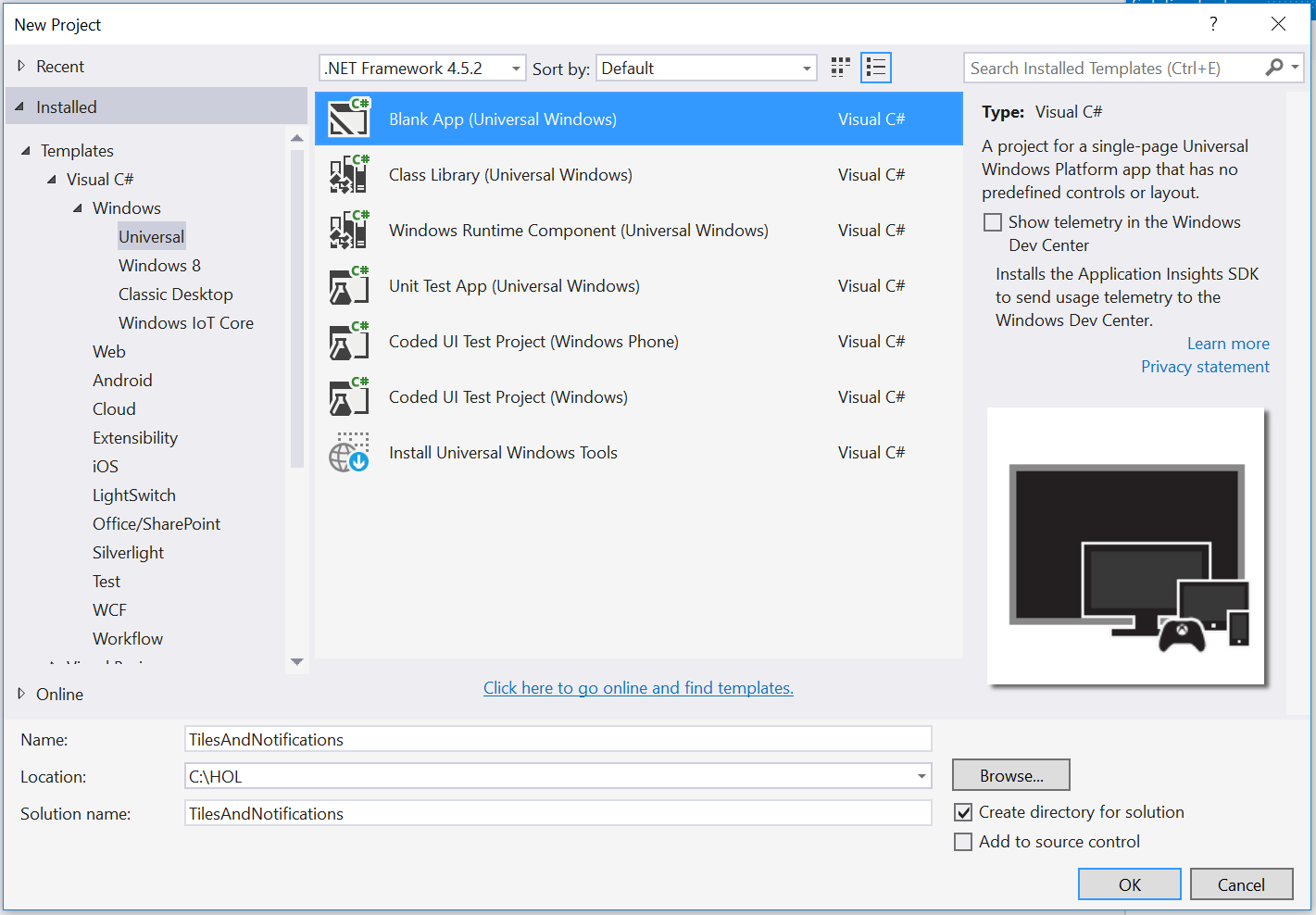


Рисунок 1

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 **DebugSettings.EnableFrameRateCounter** как **False (Ложное)**.

1. Вернитесь к Visual Studio и отключите отладку.

Задача 2 – Импортировать визуальные активы

При создании нового проекта в папке Assets создаются общие активы плитки. Чтобы отметить свое приложение, вы можете заменить общие активы, представленные в шаблоне Blank App, на собственные логотипы. Визуальные активы могут иметь место в соответствии с различными масштабными коэффициентами относительно плотности пикселей экрана. Если вы используете изображения только в соответствии с одним масштабным коэффициентом, то рекомендуется применять изображение в масштабе 200 (200% от исходного размера на стандартном устройстве с 96 dpi), что должно привести к хорошим результатам, поскольку Windows масштабирует изображения с целью идеального размещения относительно плотности экранов различных устройств. Вы можете представить изображения, выполненные при нескольких масштабных коэффициентах, для достижения наилучших результатов при различной плотности экрана.

В рамках данной задачи вы сможете добавить активы отмеченных изображений при масштабе 200 в приложение с целью их отображения на плитках по умолчанию.

1. Откройте **Package.appxmanifest** в редакторе манифестов и перейдите на вкладку **Visual Assets**. Выберите активы из **Tile Images and Logos**  в панели активов изображений.

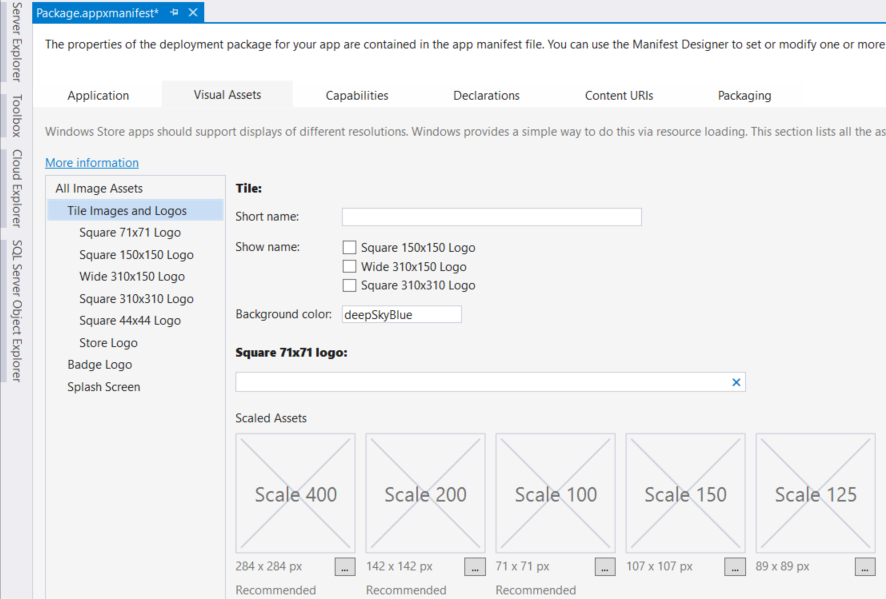


Рисунок 4

Выбирая Tile Images and Logos во вкладке Visual Assets редактора манифестов.

1. В поле Background color (Цвет фона) введите **"deepskyblue"**.
   1. **Примечание:** Если оставить поле Background color, настроенным на прозрачный, ваши плитки будут иметь выбранный ранее пользователем цвет, который пользователь затем может изменить через **Windows 10 Settings (Настройки Windows 10) > Personalization (Персонализация) > Colors (Цвета)**. В этом упражнении мы установим цвет фона для плиток таким образом, чтобы можно было легче увидеть файлы-изображения, выгруженные в редакторе манифестов.
2. Используйте символ эллипса под изображением **"Square71x71Logo"** при **шкале 200**, чтобы открыть диалоговое окно **Select Image (Выбрать изображение)**. Откройте папку **Assets** и выберите файл-изображение **"Square71x71Logo.scale-200.png"**. Нажмите **Open** для добавления изображения в качестве визуального актива.

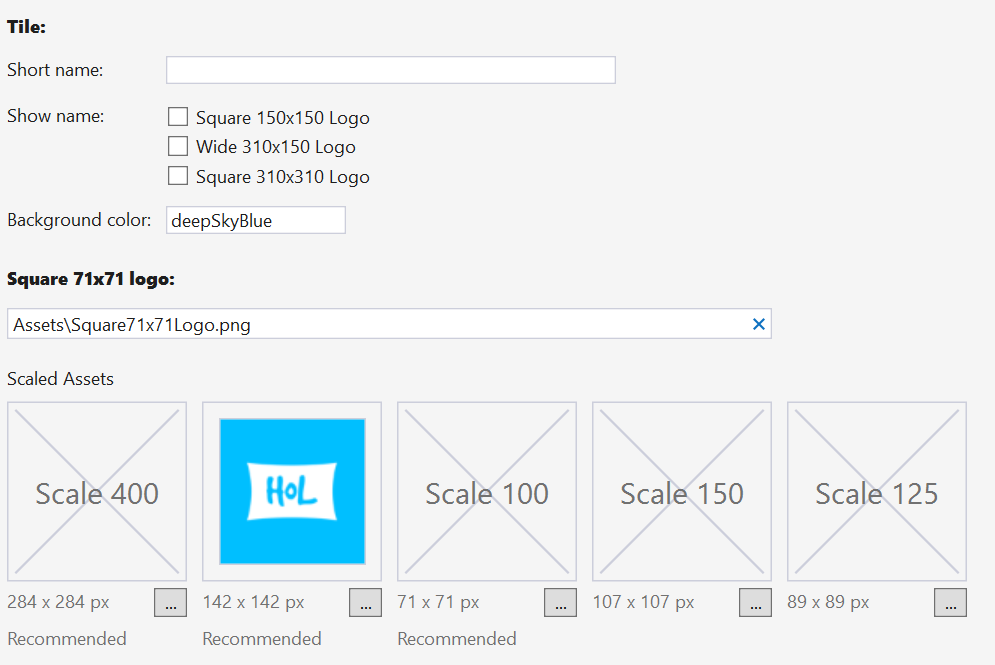


Рисунок 5

Актив "Square71x71Logo" в редакторе манифестов.

1. Повторите **Шаг 2**, чтобы добавить **"Square150x150Logo.scale-200.png"**, **"Square310x310Logo.scale-200.png"**, **"Wide310x150Logo.scale-200.png"**, **"SplashScreen.scale-200.png", "Square44x44Logo.scale-200.png"** и **"StoreLogo.png"** в ваши визуальные активы. В каждом случае при замене одного из общих активов плитки уже в самом проекте, Visual Studio выдаст: «Хотите ли вы заменить это?». Нажмите Yes.
2. Создайте и запустите свое приложение. Как только окно развернется, найдите свое приложение в Start menu (Начальном меню). Щелкните правой кнопкой мыши по наименованию приложения и выберите **Pin to Start (Закрепить на начальном экране)**.



Рисунок 6

Закрепите свое приложение в Start menu (Начальном меню).

1. Ваша плитка по умолчанию появится в Start menu с активами логотипов и цветом фона, который вы выбрали в редакторе манифестов. Щелкните правой кнопкой мыши на плитку и используйте опцию **Resize (Изменить размер)**, чтобы увидеть плитку в различных размерах.

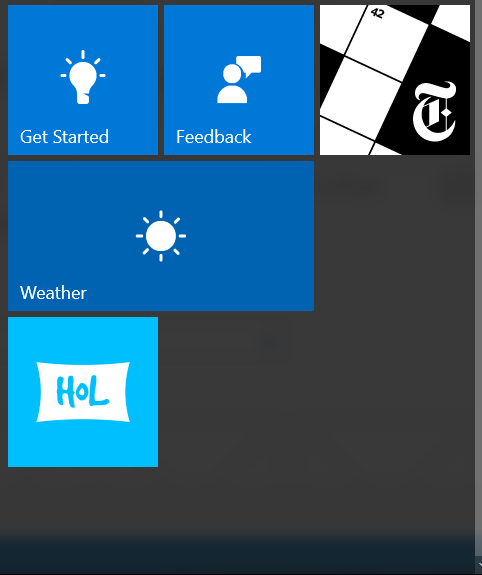


Рисунок 7

Закрепите свое приложение в Start menu (Начальном меню).

1. Отключите отладку и вернитесь в Visual Studio.

Задача 3 – Обновить счетчик уведомления плитки

Счетчик уведомления плитки является отличным способом для отображения информации на экран Glance Screen, в частности, для подсчета новых элементов. Вы можете обновить счетчик уведомлений для плитки по умолчанию, что представляет собой быстрый и легкий способ в плане добавления дополнительной информации на начальном экране.

1. Щёлкните правой кнопкой мыши по наименованию проекта и выберите **Add (Добавить) > New Folder (Новая папка)**. Назовите папку **"Services"**.
2. Щелкните правой кнопкой мыши по папке Services (Службы) и выберите **Add (Добавить) > Class (Класс)**. Назовите класс **"TileService.cs"**.
3. Откройте **TileService.cs**, сделав его **открытым классом**.
   * 1. C#
   1. namespace TilesAndNotifications.Services
   2. {
   3. public class TileService
   4. {
   5. }
4. }
5. Добавьте статический метод, чтобы обновить счетчик уведомлений на плитке. Добавьте, используя операторы для Windows.Data.Xml.Dom и Windows.UI.Notifications.
   * 1. C#
   1. using Windows.Data.Xml.Dom;
   2. using Windows.UI.Notifications;
   3. namespace TilesAndNotifications.Services
   4. {
   5. public class TileService
   6. {
   7. static public void SetBadgeCountOnTile(int count)
   8. {
   9. // Update the badge on the real tile
   10. XmlDocument badgeXml = BadgeUpdateManager.GetTemplateContent(BadgeTemplateType.BadgeNumber);
   11. XmlElement badgeElement = (XmlElement)badgeXml.SelectSingleNode("/badge");
   12. badgeElement.SetAttribute("value", count.ToString());
   13. BadgeNotification badge = new BadgeNotification(badgeXml);
   14. BadgeUpdateManager.CreateBadgeUpdaterForApplication().Update(badge);
   15. }
   16. }
6. }
7. Вернитесь к MainPage.xaml. Создайте кнопку для обновления счетчика уведомлений.
   * 1. XAML
   1. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
   2. <Button Click="UpdateBadge" VerticalAlignment="Top" Margin="12">Update Badge Count</Button>
   3. </Grid>
8. Во вспомогательном коде MainPage добавьте поле **\_count** типа int с использованием метода **UpdateBadge()** для вызова TileService. Добавьте пространство имен **TilesAndNotifications.**.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. }

int \_count;

* 1. private void UpdateBadge (object sender, RoutedEventArgs e)
  2. {
  3. \_count++;
  4. TileService.SetBadgeCountOnTile(\_count);
  5. }

1. Создайте и запустите свое приложение на Local Machine (Локальном компьютере). После появлении окна, снова найдите свое приложение в Start menu (Начальном меню).
2. 
3. Рисунок 8
4. Закрепите свое приложение в Start menu (Начальном меню).
5. В запущенном приложении нажмите на кнопку **Update Badge Count (Обновить счетчик уведомлений)**. Когда вы вернетесь к своей «Живой плитке», находящейся на Start screen (Начальном экране) , то количество уведомлений, равное 1. Число уведомлений будет меняться всякий раз, когда вы вызываете метод **UpdateBadge()**.
6. Отключите отладку и вернитесь в Visual Studio.

Упражнение 2: Создание адаптивных «Живых плиток»

* 1. «Живых плитки» в Windows 10 используют адаптивные шаблоны, чтобы доставить контент, адаптированный для устройства и плотности экрана. В то время как шаблоны прежних версий все еще совместимы с «Живыми плитками», адаптивные шаблоны предоставляют вам большую свободу в плане выбора контента, желаемого для отображения на всех устройствах. Группы и подгруппы позволяют вам семантически связывать контент в пределах плитки. В настоящем упражнении вы создадите адаптивный макет и реализуете его с последующим появлением статических данных на плитке.

Задача 1 – Добавить модель

* 1. В стандартном приложении «Живая плитка» отображает имеющиеся данные по приложению. В рамках данной задачи мы создадим класс, необходимый для формирования статических данных с целью их отображения на плитках.

1. Щелкните правой кнопкой мыши по наименованию проекта и создайте папку **"Models"**.
2. Добавьте класс **PrimaryTile.cs** в папку Models (Модели).
3. Откройте класс **PrimaryTile** и сделайте его открытым. Добавьте статические данные в качестве строковых полей для определения данных, которые мы будет использовать с целью заполнения плитки.
   * 1. C#
   1. namespace TilesAndNotifications.Models
   2. {
   3. public class PrimaryTile
   4. {
   5. public string time { get; set; } = "8:15 AM, Saturday";;
   6. public string message { get; set; } = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.";
   7. public string message2 { get; set; } = " At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident.";
   8. public string branding { get; set; } = "name";
   9. public string appName { get; set; } = "HoL";
   10. }
   11. }

Задача 2 – Создать XML-плитку

Схема адаптивной плитки написана в XML. В рамках настоящей задачи вы сгенерируете XML, необходимый для отображения текстового контента от модели PrimaryTile н маленьких и средних «Живых плитках».

1. В **TileService.cs** добавьте **пространства имен:** TilesAndNotifications.Models **и** System.Xml.Linq**.**
   * 1. C#
   1. using TilesAndNotifications.Models;
   2. using System.Xml.Linq;
2. Добавьте метод **CreateTiles**, чтобы сгенерировать XML для маленьких и средних плиток.
   * 1. C#
   1. public static Windows.Data.Xml.Dom.XmlDocument CreateTiles (PrimaryTile primaryTile)
   2. {
   3. XDocument xDoc = new XDocument(
   4. new XElement("tile", new XAttribute("version", 3),
   5. new XElement("visual",
   6. // Small Tile
   7. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileSmall"),
   8. new XElement("group",
   9. new XElement("subgroup",
   10. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   11. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   12. )
   13. )
   14. ),
   15. // Medium Tile
   16. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileMedium"),
   17. new XElement("group",
   18. new XElement("subgroup",
   19. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   20. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   21. )
   22. )
   23. )
   24. )
   25. )
   26. );
   27. Windows.Data.Xml.Dom.XmlDocument xmlDoc = new Windows.Data.Xml.Dom.XmlDocument();
   28. xmlDoc.LoadXml(xDoc.ToString());
   29. return xmlDoc;
   30. }
   31. **Примечание:** Существует несколько элементов, которые вы можете включить в свою схему адаптивной плитки. Для каждого типа элементов вы можете сделать выбор в пользу одного из предварительно заданных стилей. Для ознакомления с дополнительной информацией посетите страницу: [https://msdn.microsoft.com/en-us/library/windows/apps/Mt186446.aspx.](https://msdn.microsoft.com/en-us/library/windows/apps/Mt186446.aspx).
   32. Для ознакомления с указаниями и примерами см. данный блог-пост: [Схема адаптивных плиток и документация](http://blogs.msdn.com/b/tiles_and_toasts/archive/2015/06/30/adaptive-tile-templates-schema-and-documentation.aspx).
3. Откройте MainPage.xaml. Добавьте и расположите кнопку под кнопкой Update Badge Count с целью обновления основной плитки. Мы расположим кнопки в панели StackPanel, чтобы облегчить компоновку.
   * 1. XAML
4. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
5. <StackPanel>
6. <Button Click="UpdateBadge" VerticalAlignment="Top" Margin="12">Update Badge Count</Button>
7. <Button Click="UpdatePrimaryTile" VerticalAlignment="Top" Margin="12">Update Primary Tile</Button>
8. </StackPanel>
9. </Grid>
10. Во вспомогательном коде MainPage добавьте пространства имен: **TilesAndNotifications.Models** и **Windows.UI.Notificiations**.
    * 1. C#
    1. using TilesAndNotifications.Models;
    2. using Windows.UI.Notifications;
11. Добавьте метод **UpdatePrimaryTile()** в вспомогательный код MainPage.
    * 1. C#
    1. private void UpdatePrimaryTile(object sender, Windows.UI.Xaml.RoutedEventArgs e)
    2. {
    3. var xmlDoc = TileService.CreateTiles(new PrimaryTile());
    4. var updater = TileUpdateManager.CreateTileUpdaterForApplication();
    5. TileNotification notification = new TileNotification(xmlDoc);
    6. TileNotification notification = new TileNotification(xmlDoc);
    7. }
    8. **Примечание:** Каждый раз обновляя плитку, вы на самом деле создаете новый экземпляр такой плитки. При создании плитки в приложении с помощью «живых данных» вы можете использовать самые последние данные для отображения на плитке.
12. Создайте и запустите свое приложение. Прикрепите основную плитку своего приложения к Start menu (Начальному меню), если она еще не находится там.
13. Используйте кнопку **Update Primary Tile (Обновить основную плитку)** в запущенном приложении для запуска обновления плитки. Откройте Start Menu (Начальное меню) на своем устройстве и подождите, пока вы не увидите обновление плиток по умолчанию. Измените размер плитки, чтобы увидеть то, как работают маленькие и средние плитки.

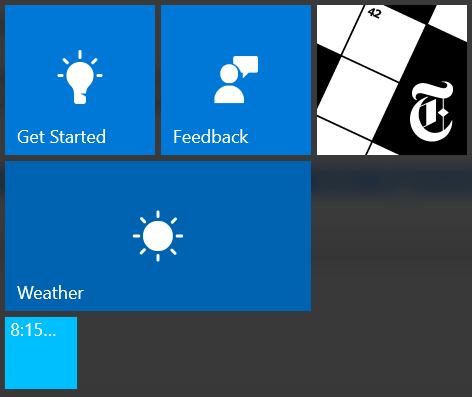


Рисунок 9

Маленькая «Живая плитка».

1. Измените размер основной плитки на **средний**. Теперь при перевороте «Живой плитки» вы сможете увидеть дополнительную информацию, предоставленную адаптивным шаблоном.

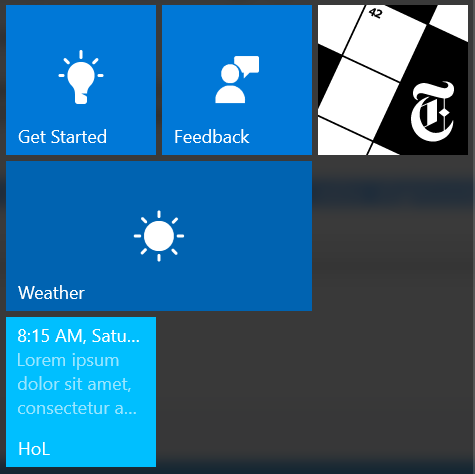


Рисунок 10

Средняя «Живая плитка».

1. Отключите отладку и вернитесь в Visual Studio.

Задача 3 – Создать адаптивные шаблоны для широких и больших плиток.

1. Добавьте широкие и большие плитки к методу **CreateTiles()**. Большие «Живые плитки» имеют пространство для отображения изображений в дополнение к тексту. В данном случае мы отобразим изображение, уже ныне существующее в папке Assets (Активы). Убедитесь, что вы добавили запятую после средней XML-плитки для продолжения списка.
   * 1. C#
   1. ),
   2. // Wide Tile
   3. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileWide"),
   4. new XElement("group",
   5. new XElement("subgroup",
   6. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   7. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   8. ),
   9. new XElement("text", primaryTile.message2, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   10. ),
   11. new XElement("subgroup", new XAttribute("hint-weight", 15),
   12. new XElement("image", new XAttribute("placement", "inline"), new XAttribute("src", "Assets/StoreLogo.png"))
   13. )
   14. )
   15. ),
   16. //Large Tile
   17. new XElement("binding", new XAttribute("branding", primaryTile.branding), new XAttribute("displayName", primaryTile.appName), new XAttribute("template", "TileLarge"),
   18. new XElement("group",
   19. new XElement("subgroup",
   20. new XElement("text", primaryTile.time, new XAttribute("hint-style", "caption")),
   21. new XElement("text", primaryTile.message, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   22. ),
   23. new XElement("text", primaryTile.message2, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true), new XAttribute("hint-maxLines", 3))
   24. ),
   25. new XElement("subgroup", new XAttribute("hint-weight", 15),
   26. new XElement("image", new XAttribute("placement", "inline"), new XAttribute("src", "Assets/StoreLogo.png"))
   27. )
   28. )
   29. )
   30. **Примечание:** Чтобы отобразить широкие и большие «Живые плитки», вы должны иметь активы WideLogo (Широкого логотипа) и Square310x310Logo (Логотипа размером 310x310), указанные в вашем манифесте приложения. Мы добавили данные активы в Упражнение 1.
2. Создайте и запустите свое приложение. Щелкните по кнопке Update Primary Tile (Обновить основную плитку). Прикрепите основную плитку и измените ее размер на **широкий** и **большой**. Имейте в виду, что при возможности отображения **message (сообщения)** и **message2 (сообщения2)** большая плитка, вероятнее всего, будет иметь пространство, необходимое для их отображения.

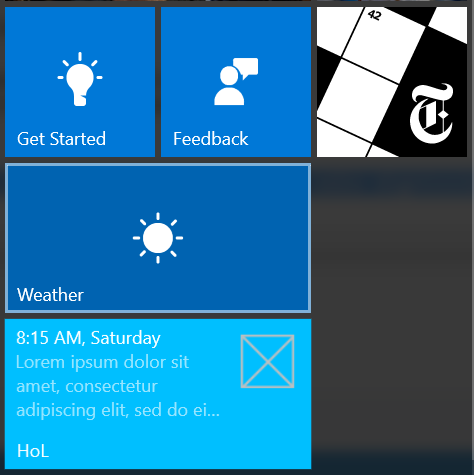


Рисунок 11

Широкая «Живая плитка».

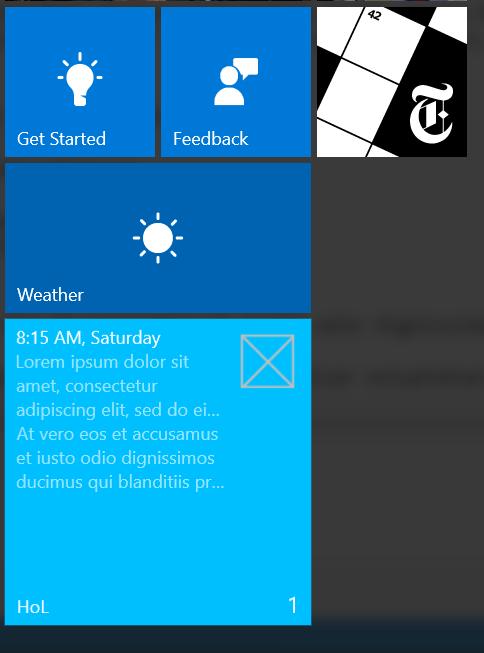


Рисунок 12

Большая «Живая плитка» имеет пространство для отображения большего количества контента.

* 1. **Примечание:** Большие плитки не доступны на устройствах, работающих на Windows 10 Mobile (мобильной версии Windows 10).

1. Отключите отладку и вернитесь в Visual Studio.

Упражнение 3: Интерактивное всплывающее уведомление (Toast)

* 1. В дополнение к адаптивным плиткам в Windows 10 обновились интерактивные и адаптивные всплывающие уведомления (Toast). Всплывающие уведомления (Toast) могут включать контент, встроенные изображения и возможность ввода данных пользователем. Приоритетные действия способствуют открытию приложения для завершения задачи в то время, пока вторичные действия реализуются без запуска приложения.
  2. В рамках данного упражнения вы создадите всплывающее уведомление (toast) с управляющими кнопками, которые помогут обновить данные в вашем приложении посредством фоновой задачи. Приложение будет содержать элемент, основу которого составляет элемент управления Checkbox и описание. Всплывающее уведомление (toast) позволит вам отметить элемент в качестве завершенного или не завершенного, ставший впоследствии видимым при обновлении главной страницы приложения.

Задача 1 – Создать службу всплывающих уведомлений (toast)

* 1. Всплывающие уведомления XML получают аналогично XML-плиткам, созданным вами в предыдущем упражнении. Мы создадим служебный класс с целью формирования XML для всплывающих уведомлений (Toast) и вызова из вспомогательного кода MainPage.

1. Щелкните правой кнопкой мыши на папку **Services (Службы)** и выберите **Add (Добавить) > Class (Класс)**. Назовите класс **"ToastService.cs"**.
2. Сделайте класс как **открытым**, так и **статичным**, а также добавьте пространства имен: **System.Xml.Linq** и **Windows.Data.Xml.Dom**.
   * 1. C#
   1. using System.Xml.Linq;
   2. using Windows.Data.Xml.Dom;
   3. namespace TilesAndNotifications.Services
   4. {
   5. public static class ToastService
3. Добавьте функцию **CreateToast()** для создания и загрузки XmlDocument.
   * 1. C#
   1. public static class ToastService
   2. {
   3. public static XmlDocument CreateToast()
   4. {
   5. var xDoc = new XDocument(
   6. new XElement("toast",
   7. new XElement("visual",
   8. new XElement("binding", new XAttribute("template", "ToastGeneric"),
   9. new XElement("text", "To Do List"),
   10. new XElement("text", "Is the task complete?")
   11. ) // binding
   12. ), // visual
   13. new XElement("actions",
   14. new XElement("action", new XAttribute("activationType", "background"),
   15. new XAttribute("content", "Yes"), new XAttribute("arguments", "yes")),
   16. new XElement("action", new XAttribute("activationType", "background"),
   17. new XAttribute("content", "No"), new XAttribute("arguments", "no"))
   18. ) // actions
   19. )
   20. );
   21. var xmlDoc = new XmlDocument();
   22. xmlDoc.LoadXml(xDoc.ToString());
   23. return xmlDoc;
   24. }
   25. }

Данный код создает следующее определение всплывающих уведомлений (Toast) за счет использования новой адаптивной XML-схемы:  
  
<toast>  
 <visual>  
 <binding template="ToastGeneric">   
<text>To Do List</text>  
 <text>Is the task complete?</text>  
 </binding>  
 </visual>  
 <actions>  
 <action activationType="background" content="Yes" arguments="yes" />  
 <action activationType="background" content="No" arguments="no" />   
</actions>  
</toast>

* 1. **Примечание:** На данный момент нет необходимости в использовании **управляющих** кнопок, однако позднее они понадобятся при выполнении упражнения. **ActivationType** **фона** подразумевает, что данные действия не приведут к запуску приложения. Для ознакомления с более подробной информацией о XML-схеме всплывающих уведомлений (Toast) посетите следующую страницу: <https://msdn.microsoft.com/en-us/library/windows/apps/br230849.aspx>

1. Сохраните и закройте ToastService.
2. Откройте **MainPage.xaml**. Добавьте класс RowDefinitions, применяемый к элементам Grid со второй панелью StackPanel в нижней строке.
   * 1. XAML
   1. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
   2. <Grid.RowDefinitions>
   3. <RowDefinition />
   4. <RowDefinition />
   5. </Grid.RowDefinitions>
   6. <StackPanel Grid.Row="0" Margin="12">
   7. <TextBlock Text="Adaptive Tiles" FontSize="20" FontWeight="Light" />

<Button Click="UpdateBadge" VerticalAlignment="Top" Margin="12">Update Badge Count</Button>

* 1. <Button Click="UpdatePrimaryTile" VerticalAlignment="Top" Margin="12">Update Primary Tile</Button>

</StackPanel>

* 1. <StackPanel Grid.Row="1" Margin="12">
  3. </StackPanel>
  4. </Grid>

1. Добавьте заголовок раздела и кнопку Notify (Уведомить) к новой панели StackPanel. На следующем этапе вы создадите метод Notify().
   * 1. XAML
   1. <StackPanel Grid.Row="1" Margin="12">
   2. <TextBlock Text="Interactive Toast" FontSize="20" FontWeight="Light" />
   3. <Button Click="Notify" Margin="12">Notify</Button>
   4. </StackPanel>
2. Добавьте пространства имен **Windows.UI.Notifications** и **TilesAndNotifications.Services** во вспомогательном коде MainPage **.**
   * 1. C#
   1. using TilesAndNotifications.Services;
3. Создайте метод **Notify()** во вспомогательном коде.
   * 1. C#
   1. private void Notify(object sender, RoutedEventArgs e)
   2. {
   3. var xmlDoc = ToastService.CreateToast();
   4. var notifier = ToastNotificationManager.CreateToastNotifier();
   5. var toast = new ToastNotification(xmlDoc);
   6. notifier.Show(toast);
   7. }
4. Создайте и запустите свое приложение на Local Machine (Локальном компьютере). Используйте кнопку **Notify (Уведомить)** для отправки всплывающего уведомления (Toast). При появлении всплывающего уведомления (toast) вы можете щелкнуть по управляющим кнопкам, но ничего не произойдет. Мы осуществим данные действия в последующей задаче.

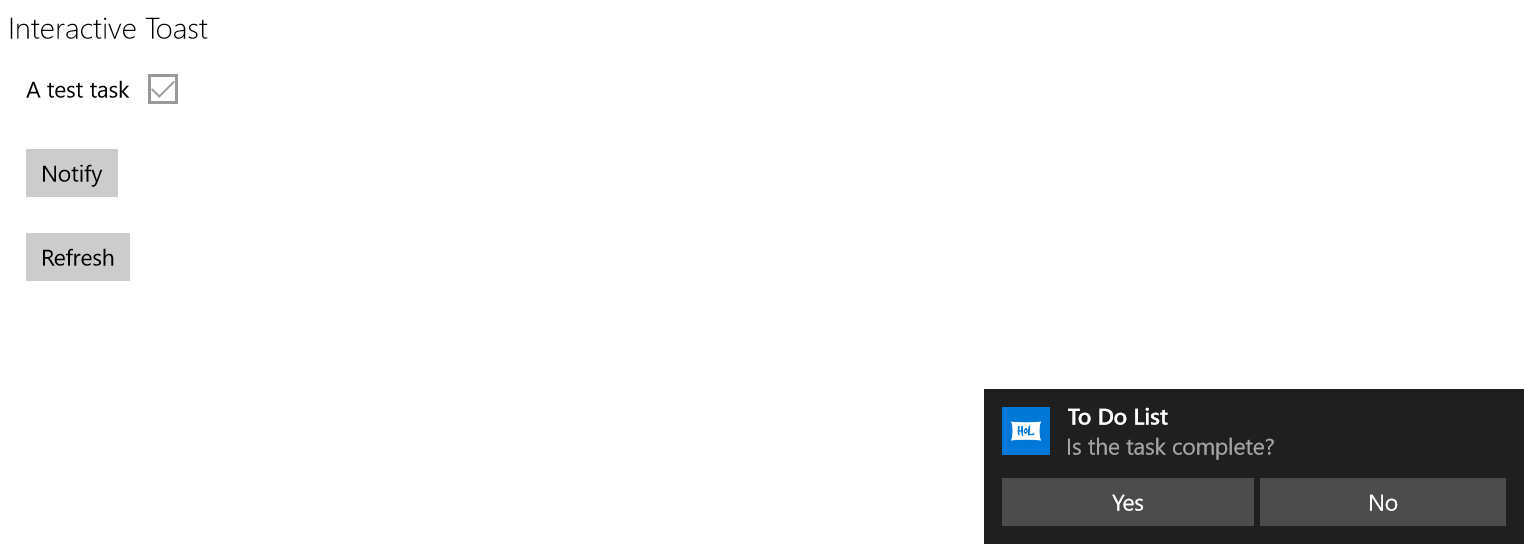


Рисунок 13

Всплывающее уведомление (toast) с управляющими кнопками.

1. Отключите отладку и вернитесь в Visual Studio.

Задача 2 – Создать модель и помощника

До того, как мы сможем реализовать всплывающие уведомления (toast), нам понадобится simple class (простой класс) и помощник. Мы создадим их в библиотеке классов, делая их доступными из любого проекта. Как приложение, так и фоновая задача, которая будет существовать в рамках компонента Windows Runtime, будут требовать доступа к модели.

1. Щёлкнитесь правой кнопкой мыши на Solution Explorer (Обозреватель решений). Затем Add (Добавить) > New Project (Новый проект). Выберите тип проекта: **Visual C# > Windows > Universal > Class Library (Universal Windows)**. Назовите его **"TasksAndNotifications.Library"**.
2. Удалите **Class1.cs**, используя контекстное меню Solution Explorer (Обозревателя решений). Если это необходимо, подтвердите, что вы хотите удалять его на постоянной основе.
3. Щелкните правой кнопкой мыши на проект **TilesAndNotifications.Library** и выберите **Add (Добавить) > Existing Item (Существующий элемент)**. Найдите папку **Lab Assets**, а также выберите **ToDoTask.cs** и **ToDoTaskFileHelper.cs**. Добавьте их в свою библиотеку классов.
4. Откройте **ToDoTask.cs**. Вы заметите, что ToDo item (Список задач) состоит из **ID**, **описания** и флага **IsComplete**. Существует два способа в классе, способствующие управлению сериализацией JSON.
   1. **Примечание:** Мы будем хранить данные из ToDo (Списка задач) в виде файла в формате JSON для того, чтобы сделать их доступными для фоновой задачи, которая может иметь доступ к файлам в пакете приложений. Позднее в рамках данного упражнения вы создадите фоновую задачу.
5. Откройте **ToDoTaskFileHelper.cs**. Данный помощник сохраняет сериализованный JSON в виде файла и вновь обращается к нему по необходимости.
6. Вернитесь к своему приложению в Solution Explorer (Обозревателе решений). Щелкните правой кнопкой мыши на папку **References (Ссылки)** и добавьте **TilesAndNotifications.Library** как ссылку.
7. Щелкните правой кнопкой мыши на папку **Assets (Активы)** и выберите **Add (Добавить) > Existing Item (Существующий элемент)**. Найдите папку Lab Assets (Лабораторные активы) и добавьте файл-стартер **task.json**.
8. Щелкните правой кнопкой мыши на файл **task.json** и откройте его **Properties (Свойства)**. Установите параметр **Build Action** на **Content**, а также сделайте так, чтобы **процесс копирования (Copy) осуществлялся в Output Directory (Выходную папку)** на постоянной основе – **Copy always**. Закройте панель Properties (Свойства).

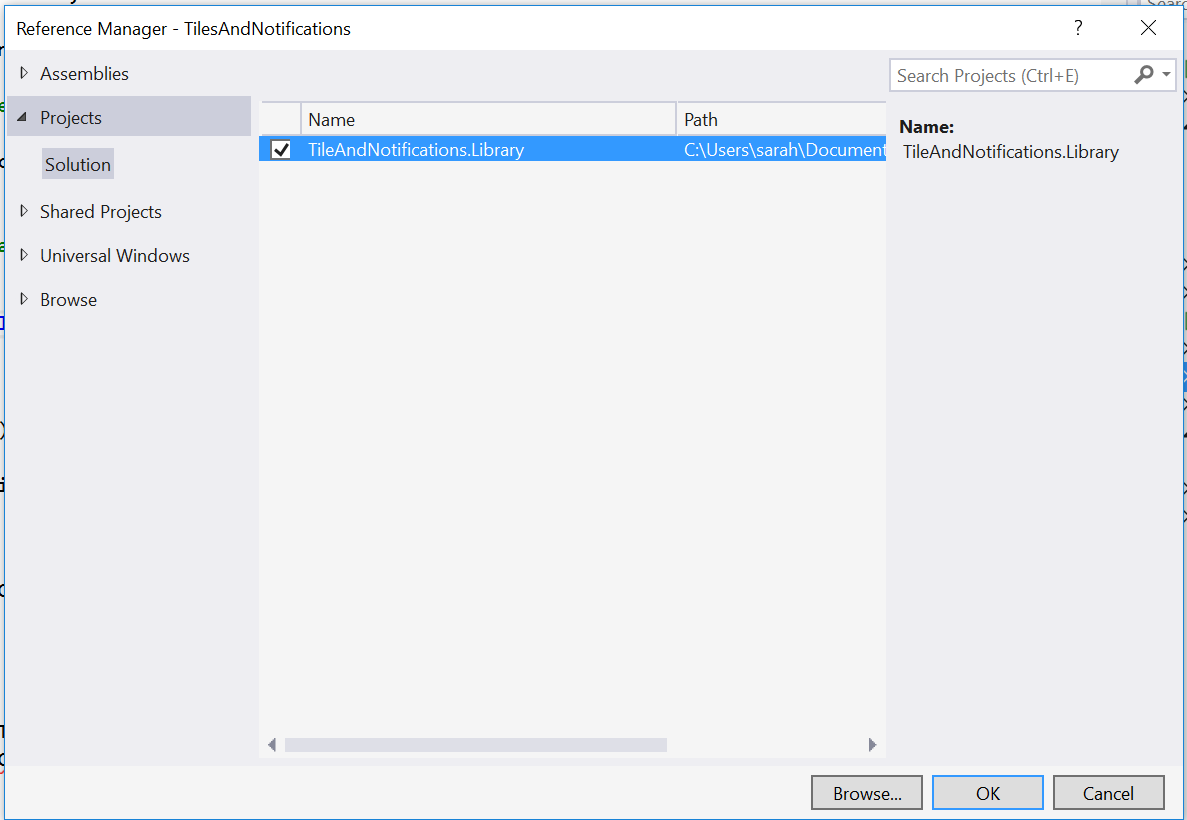


Рисунок 14

Добавьте библиотеку классов в качестве ссылке.

Задача 3 – Отобразить ToDo item (Список задач) и реализовать интерфейс INotifyPropertyChanged

Мы отобразим ToDo item (Список задач) на MainPage (Главной странице) с помощью элемента управления Checkbox для указания статуса IsComplete. ToDo task (Список задач) будет связан с соответствующим свойством во вспомогательном коде. Позднее в рамках упражнения с целью осуществления уведомления UI об изменении ToDo task посредством фоновой задачи, мы реализуем **INotifyPropertyChanged** для CurrentToDoTask.

1. Откройте **MainPage.xaml.cs**. Добавьте **пространства имен:** TilesAndNotifications.Library **и** System.ComponentModel**.**
   * 1. C#
   1. using TilesAndNotifications.Library;
   2. using System.ComponentModel;
2. Добавьте закрытое поле для ToDoTask (Списка задач) и открытое свойство класса для соответствующего доступа.
   * 1. C#
   1. private int \_count;
   2. private ToDoTask \_currentToDoTask;
   3. public MainPage()
   4. {
   5. InitializeComponent();
   6. Loaded += MainPage\_Loaded;
   7. }
   8. public ToDoTask CurrentToDoTask
   9. {
   10. get { return \_currentToDoTask; }
   11. set
   12. {
   13. \_currentToDoTask = value;
   14. }

}

1. Реализуйте **INotifyPropertyChanged** с целью уведомления UI о моменте изменения **CurrentToDoTask**. Мы коснемся UI на следующем этапе.
   * 1. C#
   1. public sealed partial class MainPage : Page, INotifyPropertyChanged

{

…

* 1. public ToDoTask CurrentToDoTask
  2. {
  3. get { return \_currentToDoTask; }
  4. set
  5. {
  6. \_currentToDoTask = value;
  7. PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(CurrentToDoTask)));
  8. }

}

public event PropertyChangedEventHandler PropertyChanged;

}

* 1. **Примечание:** INotifyPropertyChanged предоставляет уведомления об изменении общих свойств клиентам, которые обычно связаны с меняющимся значением. В этой демо-версии для UI необходима информация относительно того, когда фоновая задача при всплывающих уведомлениях (toast) способствует обновлению ToDo item (Списка задач). Для ознакомления с более подробной информацией о INotifyPropertyChanged посетите страницу: <https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.data.inotifypropertychanged>

1. Откройте MainPage.xaml и добавьте элементы управления TextBlock и CheckBox для отображения ToDo task (Списка задач) и его статуса. Добавьте кнопку для обновления данных. Вы сможете воспользоваться данной кнопкой на следующем этапе.
   * 1. XAML
   1. <StackPanel Grid.Row="1" Margin="12">
   2. <TextBlock Text="Interactive Toast" FontSize="20" FontWeight="Light" />
   3. <StackPanel Orientation="Horizontal" Margin="12">
   4. <TextBlock x:Name="Description" VerticalAlignment="Center" Text="{x:Bind CurrentToDoTask.Description, Mode=OneWay}"/>
   5. <CheckBox Margin="12,0,0,0" IsChecked="{x:Bind CurrentToDoTask.IsComplete, Mode=OneWay}" IsEnabled="False" />
   6. </StackPanel>
   7. <Button Click="Notify" Margin="12">Notify</Button>
   8. <Button Click="{x:Bind Refresh}" Margin="12">Refresh</Button>
   9. </StackPanel>
2. Вернитесь к вспомогательному коду MainPage. Добавьте async-метод **Refresh()** для считывания самых последних данных о задачах из файла JSON.
   * 1. C#
   1. private async void Refresh()
   2. {
   3. var json = await ToDoTaskFileHelper.ReadToDoTaskJsonAsync();
   4. CurrentToDoTask = ToDoTask.FromJson(json);
   5. }
3. Подпишитесь на Загруженное событие и вызовите метод Refresh() при загрузке страницы.
   * 1. C#
   1. public MainPage()
   2. {
   3. InitializeComponent();
   4. Loaded += MainPage\_Loaded;
   5. }
   6. …
   7. private void Notify(object sender, RoutedEventArgs e)
   8. {
   9. Refresh();
   10. }
   11. private async void Refresh()
   12. {
   13. var json = await ToDoTaskFileHelper.ReadToDoTaskJsonAsync();
   14. CurrentToDoTask = ToDoTask.FromJson(json);
   15. }
4. Создайте и запустите свое приложение. Задача теста появляется с выбранным по умолчанию элементом управления Checkbox – IsComplete.



Рисунок 15

ToDo task item (Список задач).

1. Отключите отладку и вернитесь в Visual Studio.

Задача 4 – Создать фоновую задачу

В рамках данной задачи вы создадите фоновую задачу как часть компонента Windows Runtime Component.

1. Щелкните правой кнопкой мыши на Solution Explorer (Оборзреватель решений). Затем Add (Добавить) > New Project (Новый проект). Добавьте проект, выполнив ряд действий:**Visual C# > Windows > Universal > Windows Runtime Component (Universal Windows)**. Назовите его **"BackgroundTasks"**.
2. Щелкните правой кнопкой мыши на папку **References (Ссылки)** в проекте BackgroundTasks и выберите **Add (Добавить) > Reference (Ссылка)**. Добавьте проект **TilesAndNotifications.Library** в качестве ссылки. Библиотека предоставит вам доступ к ToDoTask (Списку задач) и соответствующему помощнику.
3. Щелкните правой кнопкой мыши и **переименуйте** Class1.cs в **ToastUpdateTask.cs**. При необходимости согласитесь выполнить переименование всех ссылок на **Class1** в **ToastUpdateTask**.
4. Откройте **ToastUpdateTask.cs** и добавьте следующие пространства имен**:** Windows.ApplicationModel.Background**,** Windows.UI.Notifications **и** TilesAndNotifications.Library**.**
   * 1. C#
   1. using Windows.ApplicationModel.Background;
   2. using Windows.UI.Notifications;
   3. using TilesAndNotifications.Library;
5. Реализуйте интерфейс **IBackgroundTask**. Создайте и завершите задержку посредством применения метода Run (Запуска).
   * 1. C#
   1. namespace BackgroundTasks
   2. {
   3. public sealed class ToastUpdateTask : IBackgroundTask
   4. {
   5. public async void Run(IBackgroundTaskInstance taskInstance)
   6. {
   7. var deferral = taskInstance.GetDeferral();
   8. deferral.Complete();
   9. }
   10. }
   11. }
6. Преобразуйте **taskInstance.TriggerDetails** в **ToastNotificationActionTriggerDetails**. Данные детали вызываются посредством нажатия **Yes (Да)** по всплывающему уведомлению (toast). Если детали отличны от нуля, осуществите считывание JSON-задачи и установите флаг **IsComplete** на true.
   * 1. C#
   1. namespace BackgroundTasks
   2. {
   3. public sealed class ToastUpdateTask : IBackgroundTask
   4. {
   5. public async void Run(IBackgroundTaskInstance taskInstance)
   6. {
   7. var deferral = taskInstance.GetDeferral();
   8. var details = taskInstance.TriggerDetails as ToastNotificationActionTriggerDetail;
   9. if (details != null)
   10. {
   11. string arguments = details.Argument;
   12. // this is where you would retrieve any user input
   13. var userInput = details.UserInput;
   14. var json = await ToDoTaskFileHelper.ReadToDoTaskJsonAsync();
   15. var task = ToDoTask.FromJson(json);
   16. task.IsComplete = arguments == "yes";
   17. await ToDoTaskFileHelper.SaveToDoTaskJson(task.ToJson());
   18. }
   19. deferral.Complete();
   20. }
   21. }
   22. }
   23. **Примечание:** Для ознакомления с более подробной информацией об уведомлениях и иных деталей посетите страницу:<https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.notifications>
7. Задача 4 – Зарегистрировать фоновую задачу

До запуска приложения, необходимо зарегистрировать фоновую задачу в приложении TilesAndNotifications и объявить его в манифесте.

1. Вернитесь к своему проекту приложения. Откройте пакет манифеста в редакторе манифестов и перейдите на вкладу **Declarations (Манифесты)**. Выберите **BackgroundTasks** из списка **Available Declarations (Доступные манифесты)**, а затем нажмите **Add** для добавления манифеста.
2. Настройте **тип задачи на** **System Event (Системное событие)** и **Entry point (Точку входа) на BackgroundTasks.ToastUpdateTask**. Сохраните и закройте манифест.

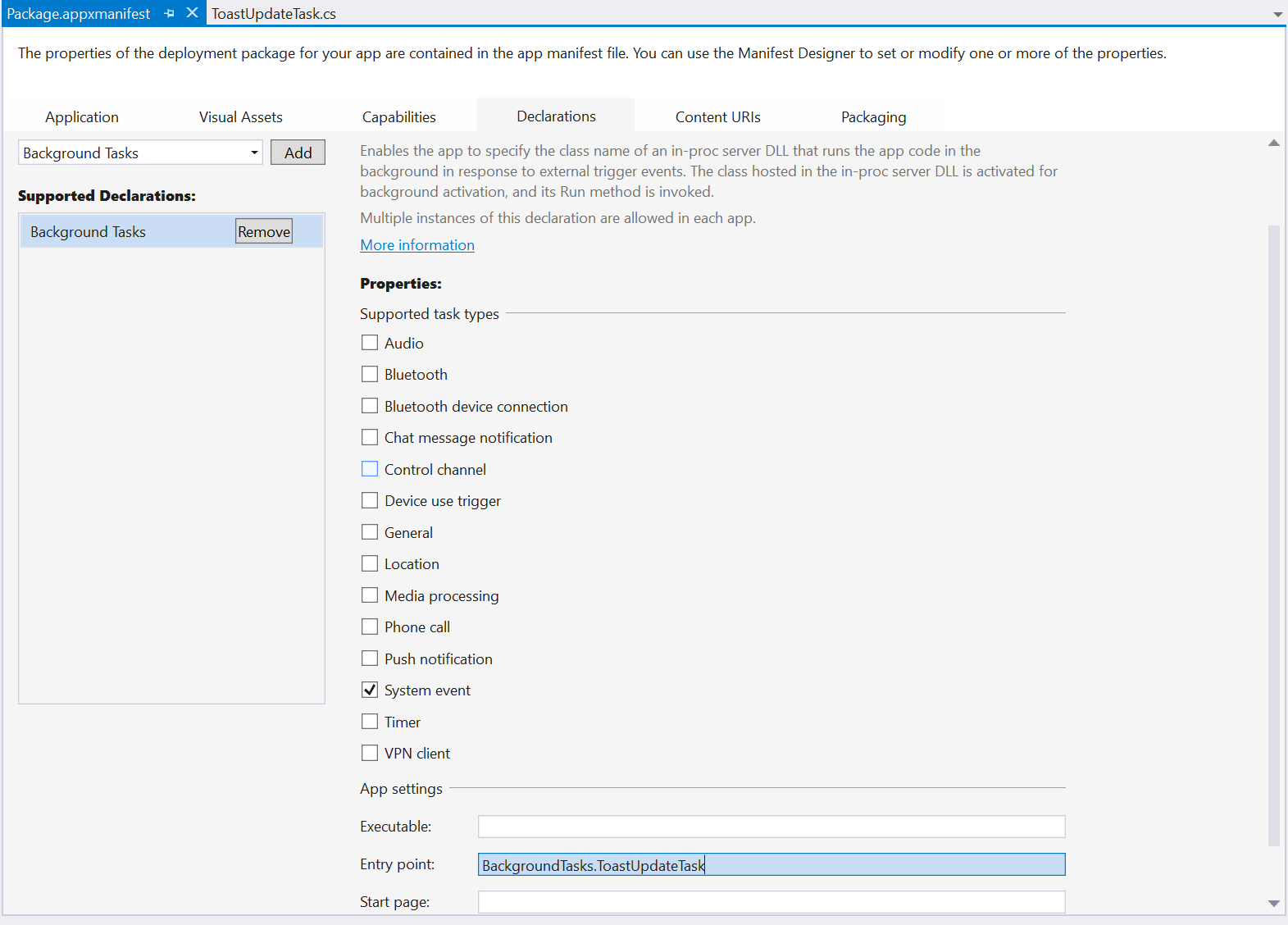


Рисунок 16

Объявите фоновую задачу в манифесте приложений.

1. В своем проекте **TilesAndNotifications** щелкните правой кнопкой мыши на папку **References (Ссылки)** и выберите **Add (Добавить) > Reference (Ссылка)**. Добавьте проект **BackgroundTasks** как ссылку.
2. Откройте **App.xaml.cs**. Добавьте **пространства имен:** Windows.ApplicationModel.Background **и** BackgroundTasks**.**
   * 1. C#
   1. using Windows.ApplicationModel.Background;
   2. using BackgroundTasks;
3. Добавьте **async**-метод под именем **RegisterBgTask** с целью управления регистрацией задач.
   * 1. C#
   1. private async void RegisterBgTask(string taskName, Type taskType, bool isInternetRequired = false)
   2. {
   3. var backgroundAccessStatus = await BackgroundExecutionManager.RequestAccessAsync();
   4. if (backgroundAccessStatus == BackgroundAccessStatus.AllowedMayUseActiveRealTimeConnectivity ||
   5. backgroundAccessStatus == BackgroundAccessStatus.AllowedWithAlwaysOnRealTimeConnectivity)
   6. {
   7. // Check to see if the task has already been registered
   8. if (BackgroundTaskRegistration.AllTasks.Any(t => t.Value.Name == taskName))
   9. {
   10. return;
   11. }
   12. // Register the toast update task
   13. var taskBuilder = new BackgroundTaskBuilder
   14. {
   15. Name = taskName,
   16. TaskEntryPoint = taskType.FullName
   17. };
   18. taskBuilder.SetTrigger(new ToastNotificationActionTrigger());
   19. var registration = taskBuilder.Register();
   20. }
   21. }
4. Вызовите **RegisterBgTask** из конструктора приложения и перейдите к типу **ToastUpdateTask**.
   * 1. C#
   1. public App()
   2. {
   3. this.InitializeComponent();
   4. this.Suspending += OnSuspending;
   5. RegisterBgTask("ToastUpdateTask", typeof(ToastUpdateTask));
   6. }
5. Создайте и запустите свое приложение. Используйте кнопку Notify (Уведомить) для отправки всплывающего уведомления (Toast). Если ваша задача была настроена на завершение, выберите **No (Нет)** в качестве основного действия. Используйте кнопку Refresh (Обновление), чтобы обновить статус задачи. Элемент управления Checkbox отобразит выбор, сделанный относительно всплывающих уведомлений (toast).

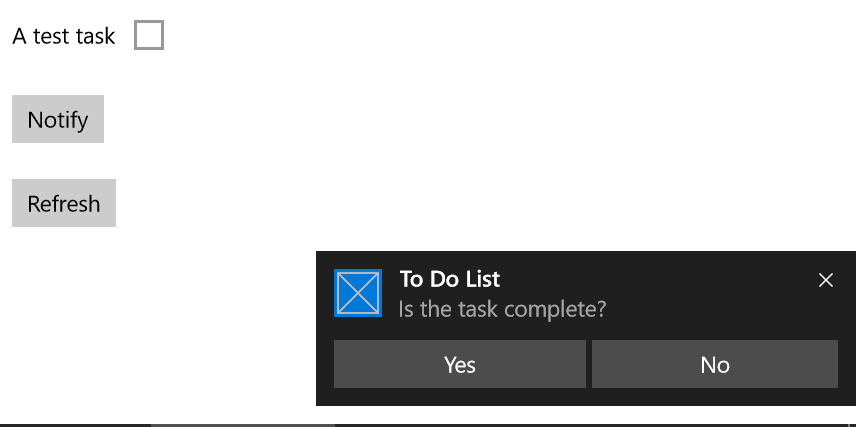


Рисунок 17

Измените статус задачи с интерактивного всплывающего уведомления (toast).

1. Отключите отладку и вернитесь в Visual Studio.

Краткий обзор

* 1. В рамках настоящего курса вы сумели добавить в свое приложение пользовательские визуальные активы и использовать их для создания разнообразных плиток по умолчанию, которые могут отображать счетчик уведомлений. Вы узнали о новой адаптивной схеме плитки и научились создавать «Живые плитки» (маленькие, средние, широкие и большие), а затем поработали над интерактивными всплывающими уведомлениями (toast) с фоновой активацией.