Windows 10

Лабораторный практикум

Навигация и кнопка "Назад"

Октябрь 2015 года

Microsoft

Обзор

В платформе UWP заложены инструменты навигации, которые призваны помочь вам обеспечить удобство при работе с любыми устройствами. В то время как приложение UWP будет выполняться любым устройством под Windows 10, необходимо планировать сценарии, в которых аппаратные средства заметно отличаются.

Класс SystemNavigationManager позволяет вам использовать программную кнопку возврата, когда аппаратная кнопка "Назад" недоступна или вы не хотите реализовывать в интерфейсе своего приложение навигацию для возврата на шаг назад, используя предоставляемые ОС решения. SystemNavigationManager также предлагает универсальное событие BackRequested, которое вы можете использовать, чтобы обработать события навигации независимо от семейства устройств или структуры пользовательского интерфейса. Эти новые возможности позволяют вам использовать возможности обратной навигации без дополнительных SDK или платформенно-зависимого кода. В приложениях под Windows 8 кнопка возврата, реализованная в интерфейсе приложения, была единственным способом возврата к предыдущему элементу интерфейса, доступным для планшетов и настольных компьютеров. Сейчас вы можете выбрать любой метод, который лучше всего работает в вашей ситуации, в зависимости от дизайна приложения и пользовательского опыта.

В этой работе мы изучим основы навигации и посмотрим, как использовать ее в простом приложении UWP. Мы поймем, как мы можем передавать данные между страницами приложения и обработаем обратную навигацию с помощью предоставленной оболочкой кнопки возврата.

Цели

Выполнив эту работу, вы сможете:

- Создать новую XAML страницу
- Перемещаться на страницы второго уровня
- Передать параметры навигации
- Использовать платформенную кнопку возврата при переполнении стека переходов назад
- Внедрять стандартный метод запроса возврата

Системные требования

Чтобы выполнить эту работу, необходимо обладать следующим набором программных инструментов:

- Microsoft Windows 10
- Microsoft Visual Studio 2015
- Эмулятор Windows 10 Mobile

Настройка

Вы должны осуществить следующие шаги для подготовки своего компьютера к этой работе:

1. Установите Microsoft Windows 10.
2. Установите Microsoft Visual Studio 2015. Выберите пользовательскую установку и убедитесь, что Инструменты разработки для приложений Windows выбраны из списка дополнительных функций.
3. Установите Windows 10 Mobile Emulator.

Упражнения

Данная работа включает следующие упражнения:

1. Навигация по странице
2. Работа с кнопкой "Назад"

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

Упражнение 1: Навигация по странице

Навигация по странице является важной частью любого приложения. В этом упражнении вы создадите вторую страницу приложения, перейдете к ней с помощью метода Frame.Navigate и научитесь передавать данные для навигации.

Задача 1 – Создание пустого приложения UWP

Начнём с создания проекта из шаблона 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 **SimpleNavigation** и выберите местоположение в файловой системе, где вы храните свои работы практического курса. 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.

   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 использовать соответствующие возможности. Нажмите **OK** для создания проекта.

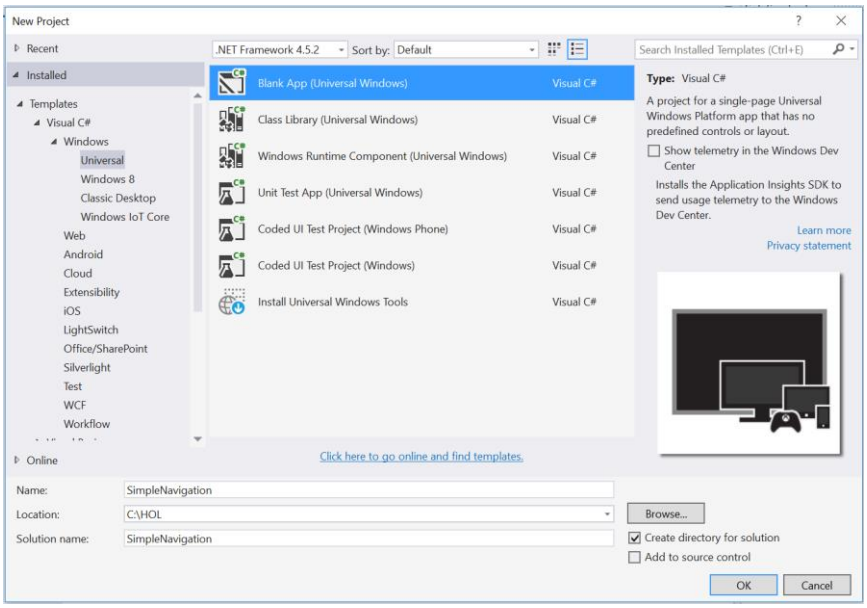

Рисунок 1

Создание нового проекта приложения в Visual Studio 2015.

3. 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).

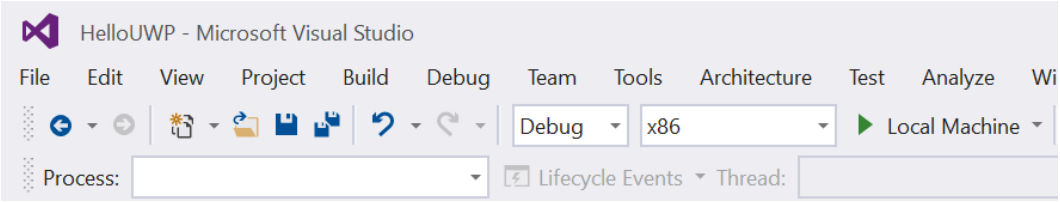

Рисунок 2

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

4. Скомпилируйте и запустите своё приложение. Вы увидите окно Blank App со счетчиком частоты кадров, активированном по умолчанию для отладки.

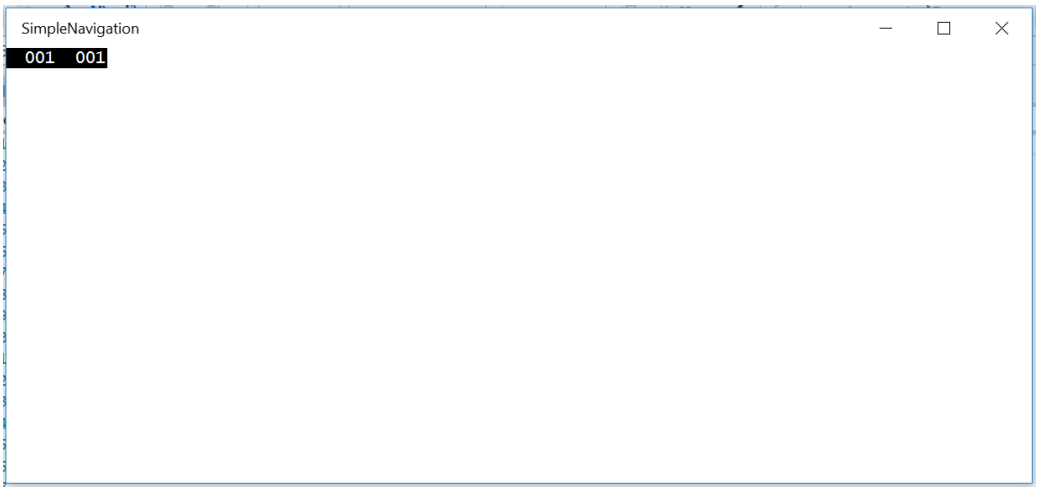

Рисунок 3

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

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 пустого приложения 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, установив значение **DebugSettings.EnableFrameRateCounter** как **False (Ложное)**.

5. Вернитесь к Visual Studio и отключите отладку.

Задача 2 – Создание новой страницы

Перед тем, как вы сможете представить навигацию, вам нужно будет создать вторую страницу, на которую вы сможете перемещаться с главной страницы приложения.

1. В вашем решении SimpleNavigation кликните правой кнопкой мыши на названии проекта в Solution Explorer (Обозреватель решения) и выберите **Add (Добавить) > New Item (Новый элемент)**.

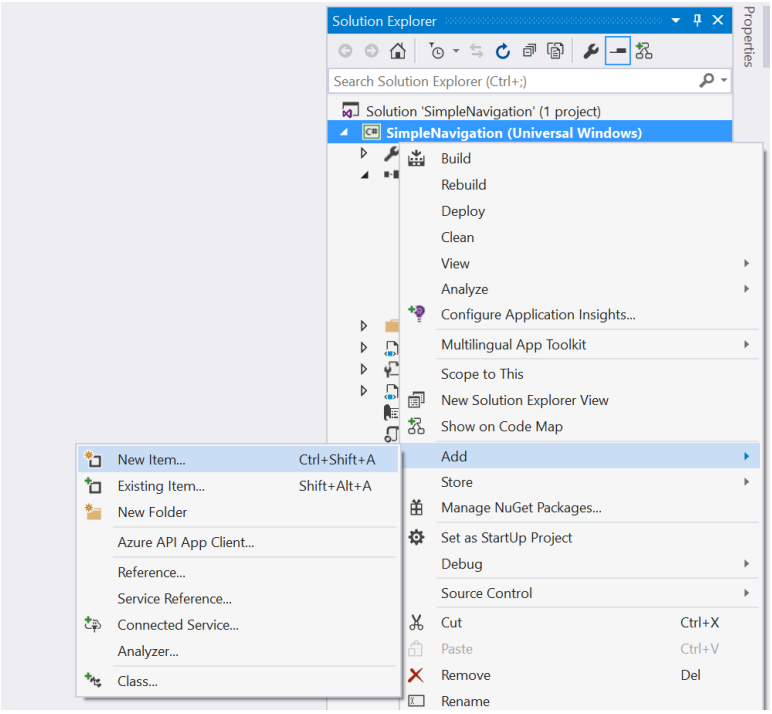

Рисунок 4

Добавление нового элемента в Solution Explorer (Обозреватель решения).

2. Выберите тип элемента **Blank Page (Чистая страница)** в списке элементов Visual C#. Назовите элемент **Page2.xaml** и кликните на кнопку **Add (Добавить)**.

> **Примечание:** Будьте внимательны и не выберите по ошибке XAML View (вид XAML)! Хотя они и похожи, чистая страница и XAML view - это не одно и то же. Чистая страница включает и файл XAML, и связанный файл с выделенным кодом. Вид XAML не включает выделенный код. Мы будем изучать вид XAML в работе, посвященной адаптивным интерфейсам.

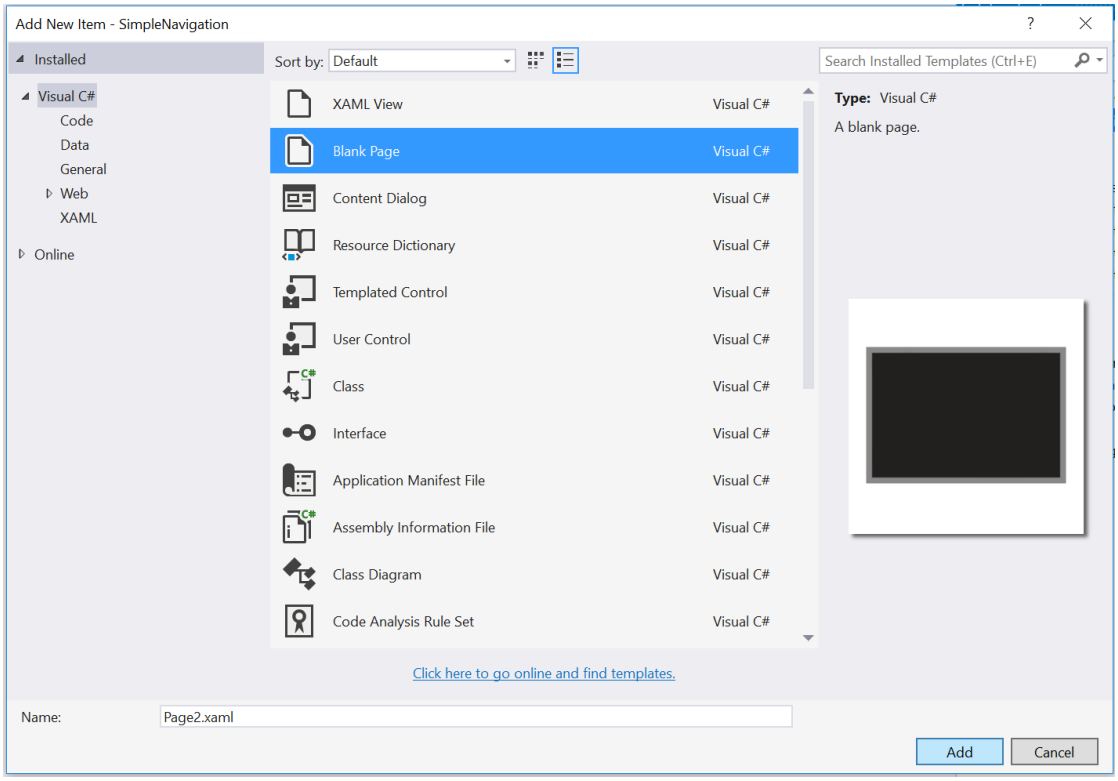

Рисунок 5

Добавление чистой страницы в проект SimpleNavigation.

3. Откройте **Page2.xaml** и добавьте текстовый блок TextBlock для отображения страничного заголовка.

XAML

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <TextBlock Text="Page 2" FontWeight="Light" FontSize="24" Margin="12"
/>
</Grid>
```

Примечание: ThemeResource - расширение для XAML разметки, которое позволяет вам ссылаться на стили XAML, определённые в словаре XAML Resource Dictionary. Расширение ThemeResource может динамически использовать другие ресурсы для отражения активной темы пользователя при запуске. Расширение StaticResource отличается тем, что оно не выполняет обновление во время запуска.

Дополнительную информацию по использованию расширения ThemeResource см. по ссылке https://msdn.microsoft.com/en-us/library/windows/apps/dn263118.aspx

4. Откройте **MainPage.xaml** и добавьте туда заголовок страницы. Мы используем простые страницы, таким образом, будет полезно знать, где мы находимся при использовании навигации.

XAML

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
```

```
    <TextBlock Text="Page Navigation" FontWeight="Light" FontSize="24" />
</Grid>
```

Задача 3 – Создание навигации

Сейчас, когда у вас есть две страницы в вашем проекте, вы можете перемещаться между ними. Навигация внутри приложения выполняется во фрейме, который служит как контейнер для ваших страниц. Когда ваше приложение запускается, корневой фрейм встраивается в App.xaml.cs и прикрепляется к окну. Фрейм важен в том смысле, что он управляет навигацией между страницами. В этой задаче вы создадите кнопку на главной странице, выполняющую переход к Странице 2.

1. Откройте **MainPage.xaml** и добавьте кнопку, которая выполняет навигацию на Страницу 2. Определите положение заголовка страницы и кнопку с помощью StackPanel, чтобы улучшить вёрстку страницы.

   XAML

   ```
   <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
       <StackPanel HorizontalAlignment="Left" Margin="12">
           <TextBlock Text="Page Navigation" FontWeight="Light" FontSize="24"
   />
           <Button Content="Go to Page 2" Margin="0,12,0,0" />
       </StackPanel>
   </Grid>
   ```

2. Добавьте событие нажатия на кнопку. Создайте метод **Button_Click**, чтобы обработать навигацию на следующем этапе.

   XAML

   ```
   <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
       <StackPanel HorizontalAlignment="Left" Margin="12">
           <TextBlock Text="Page Navigation" FontWeight="Light" FontSize="24"
   />
           <Button Content="Go to Page 2" Margin="0,12,0,0"
                   Click="Button_Click" />
       </StackPanel>
   </Grid>
   ```

3. Сейчас нужно добавить обработчик **Button_Click** в **MainPage.xaml.cs**. Когда вы вызываете **Frame.Navigate**, фрейм загружает контент указанной страницы. Он принимает в качестве параметра тип объекта страницы, для которого вы хотите разместить ссылку в навигации, а второй дополнительный параметр используется для передачи параметра. Мы передадим параметр позже в этом упражнении, сейчас пока будем обходиться без него.

C#

```
public sealed partial class MainPage : Page
{
    public MainPage()
    {
        this.InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        Frame.Navigate(typeof(Page2));
    }
}
```

4. Создайте и запустите своё приложение на локальном компьютере. Когда вы кликните по кнопке **Go to Page 2 (Перейти к странице 2)** на главной странице, ваш фрейм будет перенаправлен на страницу 2. Мы не включили кнопку "Назад", таким образом, вы пока не сможете вернуться на главную страницу. Вы научитесь обрабатывать команды для возврата к предыдущей странице в следующем упражнении.

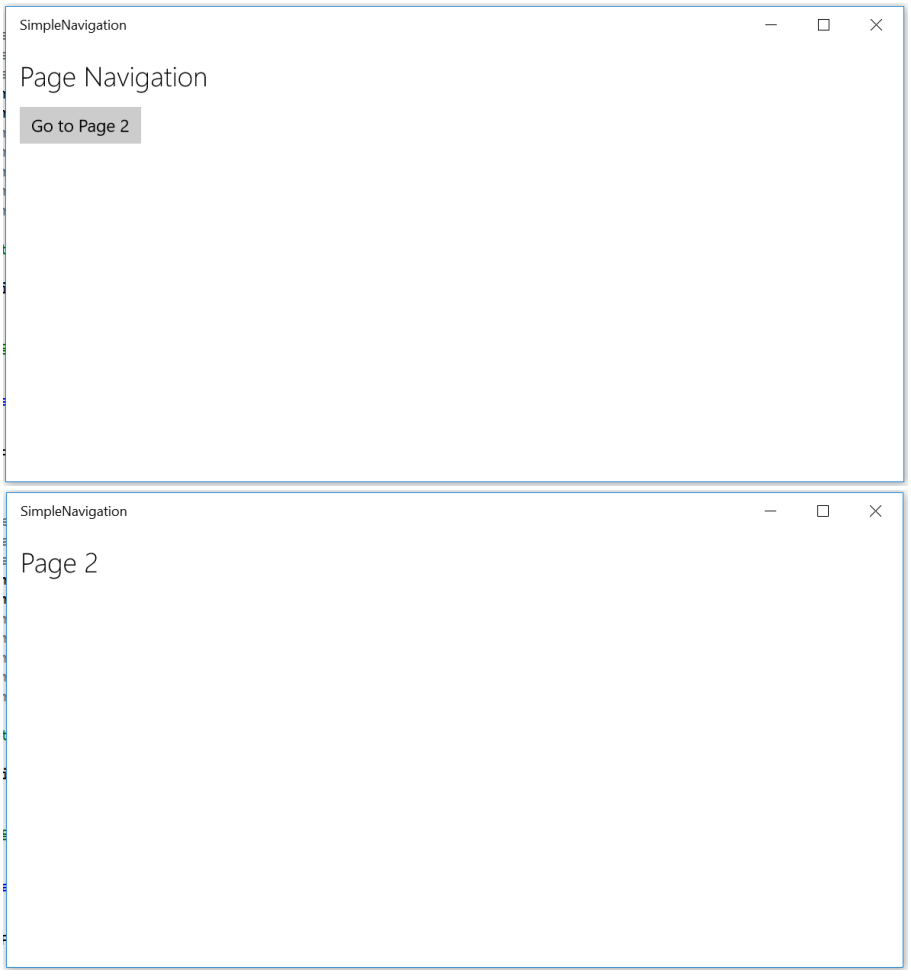

Рисунок 6

*Кнопка **Go to Page 2 (Перейти к странице 2)** позволяет осуществлять переход на другую страницу в приложении.*

5. Остановите отладку и вернитесь к Visual Studio.

Задача 4 – Передача параметра на страницу 2

Вы научились успешно перемещаться между страницами в своём приложении. Часто бывает полезно передать информацию на новую страницу при навигации. В этой задаче вы передадите параметр с главной страницы на страницу 2.

1. Добавьте текстовый блок **TextBox** в **MainPage.xaml**. Этот текстовый блок будет использоваться для ввода данных пользователями, которые вы передадите на Страницу 2 для отображения позже.

XAML

```
<StackPanel HorizontalAlignment="Left" Margin="12">
    <TextBlock Text="Page Navigation" FontWeight="Light" FontSize="24" />
    <TextBox x:Name="Message" Header="Enter a parameter to send to Page 2"
Width="300" Margin="0,12,0,0" />
    <Button Content="Go to Page 2" Margin="0,12,0,0"
            Click="Button_Click" />
</StackPanel>
```

2. В свой обработчик нажатия кнопки **Button_Click** в выделенном коде добавьте второй дополнительный параметр для **Frame.Navigate**, чтобы передать текст в TextBox на страницу 2.

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   Frame.Navigate(typeof(Page2), Message.Text);
}
```

Примечание: Дополнительный параметр, который вы передаёте в **Frame.Navigate** не обязательно должен быть строкой, допускается любой объект, но он должен быть сериализуемым. Объект, который является сериализуемым, может преобразовываться в поток байт для хранения, чтобы сохранить его состояние и воспроизвести это состояние позже. Фрейм отслеживает историю приложения и параметры навигации, которые ему нужны, чтобы возобновить работу после приостановления, и он выполняет эту задачу, сериализуя параметры. Если в более сложных случаях вы захотите передать параметр, который не является сериализуемым, вы можете написать код, чтобы сохранять его самостоятельно.

Задача 5 – Отображение сообщения, переданного на страницу 2

Ваше сообщение будет передано как параметр, когда вы перемещаетесь на страницу 2. Однако, мы пока не сделали ничего, чтобы обработать сообщение на странице 2, так что оно

пока не отобразится. Давайте добавим всплывающее диалоговое окно, которое будет отображать сообщение.

1. Откройте **Page2.xaml.cs**. Создайте переопределение для метода **OnNavigatedTo()**.

C#

```
public Page2()
{
    this.InitializeComponent();
}
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

Примечание: Конструктор страниц не будет вызываться каждый раз, когда вы переходите к странице, если страница уже была загружена. Метод **OnNavigatedTo()** вызывается каждый раз, когда вы переходите к странице, так что мы можем использовать его, чтобы запустить отображение входящего сообщения.

2. Добавьте пространство имен **Windows.UI.Popups** в выделенный код для **страницы 2**.

C#

```
using Windows.UI.Popups;
```

3. В функции **OnNavigatedTo()** входящий параметр передается в качестве аргументов NavigationEventArgs. Выведем его на экран. Поскольку функция показа диалогового окна – асинхронная, к заголовку функции надо будет добавить ключевое слово **async**.

C#

```
protected async override void OnNavigatedTo(NavigationEventArgs e)
{
    await new MessageDialog("You sent: " + e.Parameter).ShowAsync();

    base.OnNavigatedTo(e);
}
```

Примечание: Асинхронные методы позволяют приложению продолжить выполнение без ожидания операций, которые потенциально могут блокировать пользовательский интерфейс. Пользовательский интерфейс продолжит реагировать на действия пользователя, в то время как операция будет работать в фоновом режиме.

Асинхронный метод по традиции имеет имя, заканчивающееся на Async, и возвращает значение **Task** или **Task<T>**. При вызове с использованием ключевого слова **await**, рабочая среда приостанавливает содержащийся метод, и возвращает управление вызывающей программе вместе со значением задачи. Любые методы, вызывающие функции, использующие ключевое слово await, должны быть помечено с помощью ключевого слова **async**. Задача обычно выполняется асинхронно на пуле подпроцессов вместо основного потока приложения. Когда выполнение синхронного метода завершается, связанная задача

помечается как завершенная, и доступ к любым возвращаемым значениям может быть осуществлён посредством задачи.

Дополнительную информацию и примеры шаблонов асинхронного программирования см. по ссылке https://msdn.microsoft.com/en-us/library/hh191443.aspx

4. Скомпилируйте и запустите своё приложение. Наберите сообщение в текстовом поле на главной странице и перейдите к странице 2. Вы увидите, что на странице 2 появилось всплывающее окно с вашим сообщением.

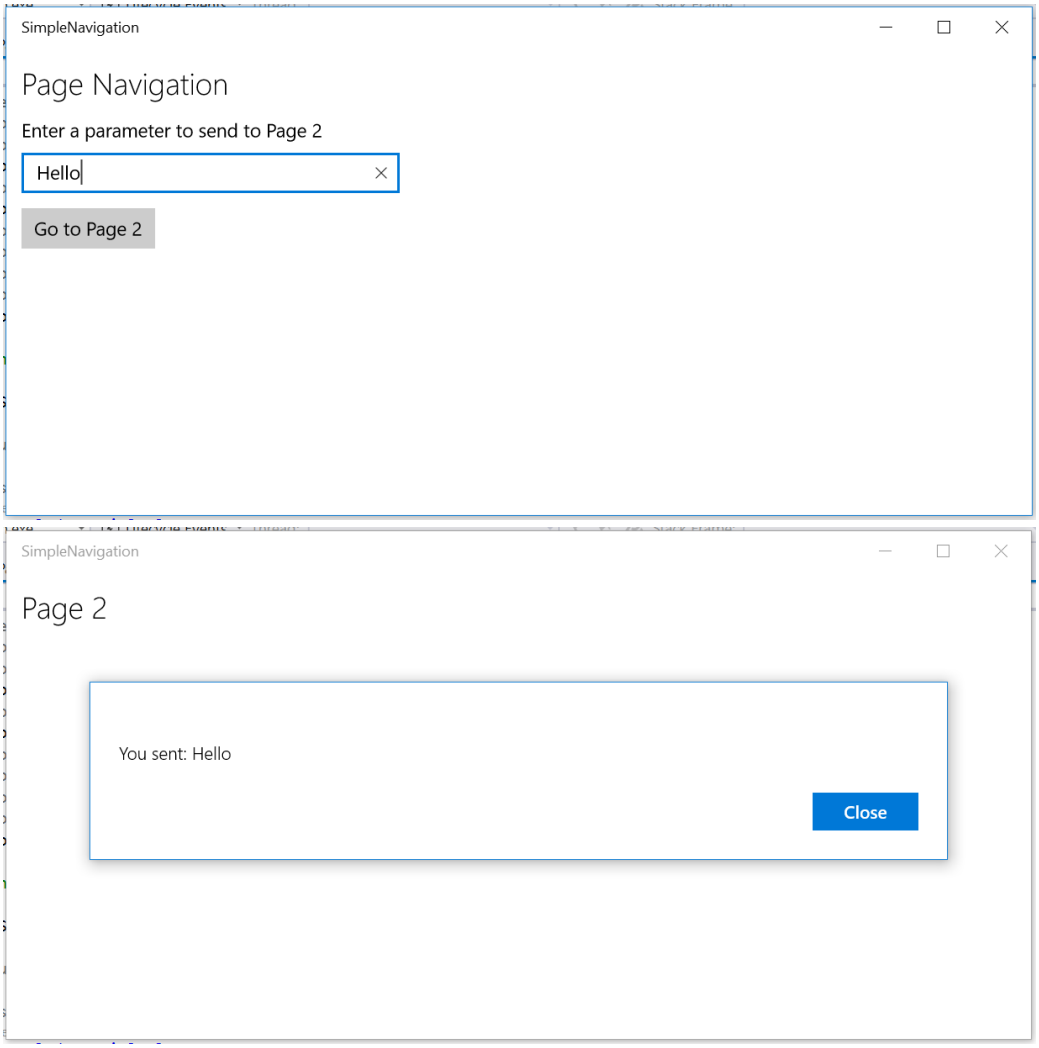

Рисунок 7

Сгенерированное пользователями сообщение передано на Страницу 2 как параметр.

5. Завершите отладку и вернитесь к Visual Studio.

Упражнение 2: Работа с кнопкой "Назад"

Приложения UWP выполняются на целом ряде устройств, которые отличаются тем, как они обращаются с обратной навигацией. Тогда как устройства под Windows 10 Mobile обычно имеют аппаратную кнопку "Назад", планшеты и настольные ПК обычно лишены подобных инструментов. Существует несколько способов обращения с возвратной навигацией внутри приложения. В этом упражнении вы изучите различия в навигации между семействами устройств и будете использовать предоставленную платформой кнопку "Назад", чтобы вернуться на главную страницу со второстепенной страницы.

Задача 1 – Проверка работы аппаратной кнопки "Назад"

Давайте посмотрим на существующий порядок работы аппаратной кнопки "Назад" на мобильном устройстве.

1. Переключите режим отладки на эмулятор мобильной платформы. Мы выбрали **Mobile Emulator 10.0.10240.0 720p 5 inch 1GB**.

2. Создайте и запустите приложение SimpleNavigation в эмуляторе. Перейдите на страницу 2. На странице 2 кликните или нажмите на аппаратную кнопку "Назад" эмулятора. Вместо того, чтобы вернуться на главную страницу, приложение завершит работу и вернёт вас в ваше последнее положение в стек переходов назад. В этом случае, если вы просто запустили эмулятор, последним положением будет экран запуска.

3. Откройте приложение Photos (Фото) с экрана запуска.

4. Кликните по аппаратной кнопке "Назад" в эмуляторе и удерживайте её, чтобы просмотреть открытые приложения. Используйте эту функцию, чтобы вернуться к своему приложению Simple Navigation.

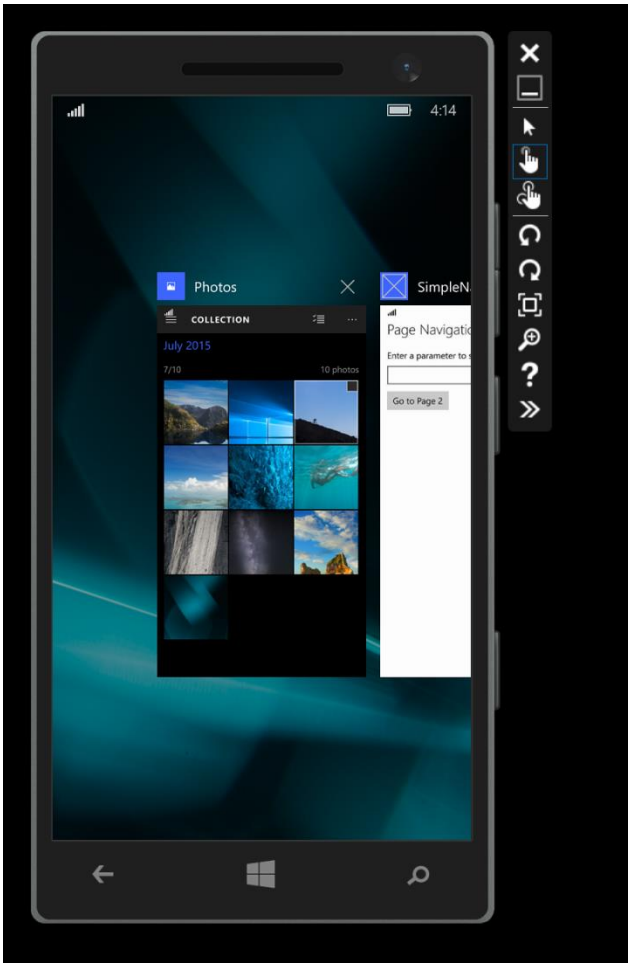

Рисунок 8

Нажмите и удерживайте кнопку "Назад", чтобы перемещаться между открытыми приложениями.

5. В приложении Simple Navigation нажмите на обратную кнопку снова. Вы вернётесь к предыдущему приложению в стеке приложений, которое является приложением Фото, а не к главной странице приложения Simple Navigation, как вы могли бы ожидать. Если вы не реализуете пользовательский возврат назад, аппаратная кнопка "Назад" будет перемещаться через стек приложений по умолчанию.
   Вы реализуете код, чтобы обращаться с пользовательским возвратом назад в **работе № 3** этого курса.

Задача 2 – Реализация программной кнопки "Назад"

Windows 10 предоставляет кнопку "Назад" в панели задач в режиме планшета, но по умолчанию кнопка "Назад" в интерфейсе операционной системы в режиме Рабочего Стола отсутствует. **SystemNavigationManager** предоставляет опции, чтобы включить метод **AppViewBackButton** для приложений, выполняющихся в режиме рабочего стола. Эта программная кнопка "Назад" отображается в панели заголовка приложения при работе в

режиме рабочего стола. Если вы напишите код для показа кнопки "Назад" операционной системой, он будет проигнорирован на мобильных устройствах, потому что операционная система ожидает наличия аппаратной кнопки "Назад", и проигнорирован при работе в режиме планшета. В этой задаче вы отобразите предоставляемую системой кнопку возврата в режиме рабочего стола, когда пользователь будет находиться на странице в приложении, где возможен возврат назад.

1. Откройте **App.xaml.cs**. Добавьте пространство имён **Windows.UI.Core**.

C#

```
using Windows.UI.Core;
```

2. Добавьте обработчик событий **rootFrame.Navigated** в конце переопределения **OnLaunched**. Этот обработчик событий будет запускать каждый раз имеющуюся навигацию на корневом фрейме и отобразит кнопку "Назад", если стек переходов назад приложения не будет пустым.

C#

```
    // Убедитесь, что текущее окно активно
    Window.Current.Activate();

    rootFrame.Navigated += RootFrame_Navigated;
}

private void RootFrame_Navigated(object sender, NavigationEventArgs e)
{
       Frame rootFrame = Window.Current.Content as Frame;

SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility
          = rootFrame.CanGoBack ? AppViewBackButtonVisibility.Visible :
                                  AppViewBackButtonVisibility.Collapsed;
};
```

Примечание: Мы хотим отображать кнопку "Назад" операционной системы, когда стек переходов назад не пустой. В этом коде устанавливается значение AppViewBackButtonVisibility по AppViewBackButtonVisibility.Visible, если значение rootFrame.CanGoBack - true, иначе используется значение AppViewBackButtonVisibility.Collapsed.

3. Измените платформу отладки на **Local Machine (Локальный компьютер)**. Создайте и запустите своё приложение и перейдите к странице 2. Кнопка "Назад" появится в панели заголовка в режиме Рабочего Стола.

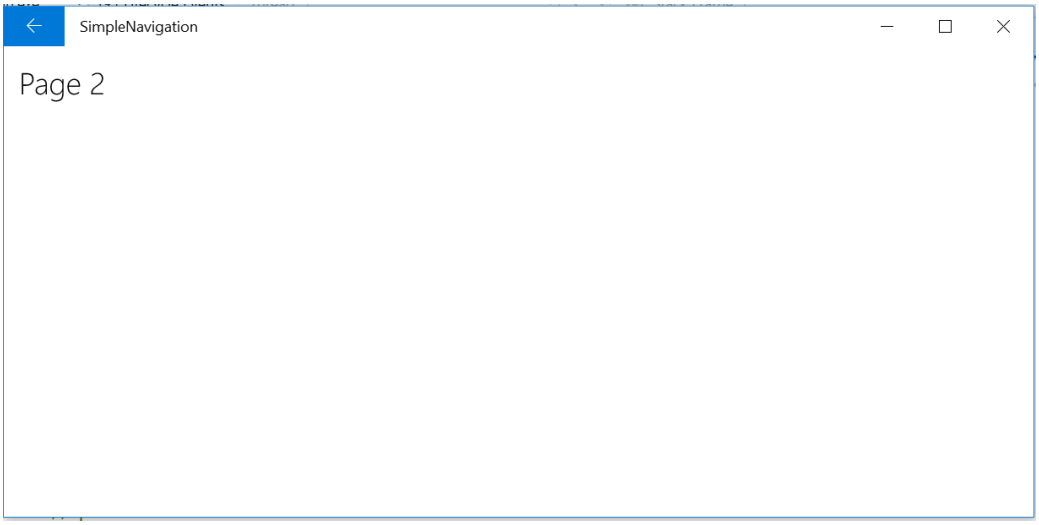

Рисунок 9

Значение AppViewBackButton - visible, кнопка отображается в панели заголовка в режиме Рабочего Стола.

4. Кликните по кнопке "Назад" в панели заголовка. Ничего не произойдёт. Хотя вы обеспечили видимость кнопки, она ещё не подключена для работы с навигацией.

Примечание: Цвет кнопки определяется выбранной пользователем цветовой схемой, которую пользователь может изменить в **Windows 10 Settings (Настройки Windows 10) > Personalization (Персонализация) > Colors (Цветовая схема)**. Если вы захотите управлять положением или стилем кнопки "Назад", можно реализовать пользовательский вариант кнопки "Назад" в самом приложении, что вы сделаете позднее в рамках данного курса.

5. Не закрывая приложение, используйте панель уведомлений Windows 10, чтобы переключиться в режим **Планшета**. Кнопка "Назад" появится в панели задач.

Рисунок 10

Кнопка "Назад" отображается в панели задач в режиме планшета.

6. Кликните по кнопке "Назад" в панели задач. Вы вернётесь в стек приложений. В отличие от кнопки в панели заголовка, кнопка "Назад" в режиме планшета демонстрирует то же поведение, что и аппаратная кнопка на мобильном устройстве. Эта кнопка отображается, даже если кнопка "Назад" оболочки в панели заголовка не включена.

Примечание: В режиме разделенного экрана существует стек переходов назад, доступный для каждой стороны экрана отдельно.

7. Используйте панель уведомлений, чтобы вернуться к режиму Настольного Компьютера. Остановите отладку и вернитесь к Visual Studio.

8. Запустите приложение снова, в этот раз в эмуляторе мобильной платформы. Когда вы будете перемещаться к странице 2, кнопка "Назад" оболочки не появится, потому что она игнорируется в режиме работы мобильной платформы.

Рисунок 11

AppViewBackButton скрыт в мобильном режиме работы.

9. Завершите отладку и вернитесь к Visual Studio.

Задача 3 – Обработка стандартного запроса на возврат назад

На данный момент мы имеем интерфейс перехода назад, работающий как под Windows 10 Mobile, так и в настольном режиме Windows 10 и в режиме планшета. Давайте рассмотрим стандартную модель запроса на переход назад, чтобы обрабатывать подобные запросы.

1. В **App.xaml.cs** создайте и подпишитесь на событие **App_BackRequested**.

Примечание: Подписка на **App_BackRequested** должна производиться только после того, как вид был создан. Если вы попытаетесь выполнить этот код до того, как вид будет создан, **SystemNavigationManager.GetForCurrentView()** будет возвращать значение **null**.

C#

```
    SystemNavigationManager.GetForCurrentView().BackRequested +=
        App_BackRequested;

    rootFrame.Navigated += RootFrame_Navigated;
}

private void RootFrame_Navigated(object sender, NavigationEventArgs e)
{
    Frame rootFrame = Window.Current.Content as Frame;
    SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility
=
        rootFrame.CanGoBack ?
           AppViewBackButtonVisibility.Visible :
           AppViewBackButtonVisibility.Collapsed;
};

private void App_BackRequested(object sender, BackRequestedEventArgs e)
{
}
```

2. В обработчике событий **App_BackRequested** проверьте, было ли событие BackRequested обработано. Если нет, установите по умолчанию перемещение назад в пределах фрейма. Убедитесь, что вы установили для **e.Handled** значение **true**, после выполнения.

C#

```
private void App_BackRequested(object sender, BackRequestedEventArgs e)
{
    // Убедитесь, что никто ещё не обработал это событие
    if (!e.Handled)
    {
        // По умолчанию установлен переход назад в пределах фрейма
        Frame frame = Window.Current.Content as Frame;
        if (frame.CanGoBack)
        {
            frame.GoBack();
```

```
            // Сигнал обработан так, что система не осуществляет переход
назад
            // через стек приложений
            e.Handled = true;
        }
    }
}
```

3. Скомпилируйте и запустите своё приложение. Перейдите к **странице 2**, затем используйте кнопку "Назад" оболочки, чтобы вернуться к главной странице.

4. Остановите отладку и вернитесь к Visual Studio.

Задача 4 – Отрисовка кнопки "Назад" в приложении

Если бы вы хотите иметь больше контроля над стилем и поведением кнопки "Назад", вы можете создать свою собственную кнопку, чтобы работать с навигацией при переходе назад.

1. Закомментируйте видимость кнопки "Назад" оболочки в **App.xaml.cs**.

C#

```
//rootFrame.Navigated += RootFrame_Navigated;
```

2. Добавьте кнопку в **Page2.xaml** и привяжите её стиль к StaticResource **NavigationBackButtonNormalStyle**. Установите положение кнопки и заголовка страницы в горизонтальное положение в StackPanel для выравнивания. Замените значение поля TextBlock заголовка на **VerticalAlignment="Top"**.

XAML

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel Orientation="Horizontal">
        <Button Style="{StaticResource NavigationBackButtonNormalStyle}"
         VerticalAlignment="Top" />
        <TextBlock Text="Page 2" FontWeight="Light"
                   FontSize="24" VerticalAlignment="Top" />
    </StackPanel>
</Grid>
```

3. Добавьте событие клика на кнопку и свяжите его с обработчиком **GoBack**. Вы создадите обработчик **GoBack** на следующем этапе.

XAML

```
<Button Style="{StaticResource NavigationBackButtonNormalStyle}"
        Click="GoBack" VerticalAlignment="Top" />
```

4. В выделенном коде страницы 2 добавьте функцию GoBack(), чтобы обработать событие перехода назад.

C#

```
private void GoBack()
{
```

```
     if (Frame.CanGoBack)
        Frame.GoBack();
}
```

5. Скомпилируйте и запустите своё приложение. Когда вы будете перемещаться к странице 2, вы увидите свою пользовательскую кнопку "Назад" вместо кнопки "Назад" оболочки. Пользовательская кнопка вернёт вас на главную страницу.

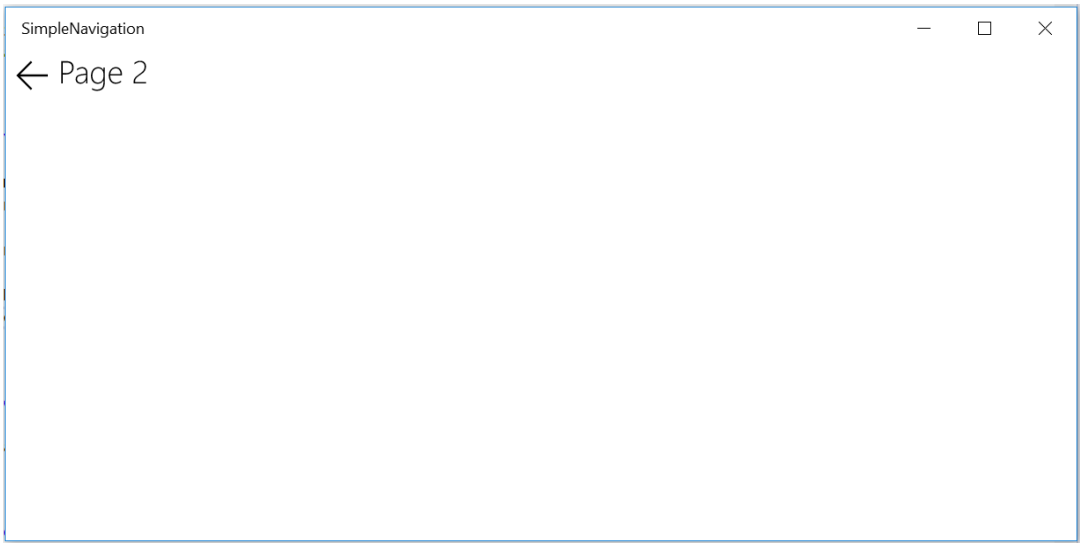

Рисунок 12

Пользовательская кнопка возврата на странице 2.

6. Завершите отладку и вернитесь к Visual Studio.

Задача 5 – Управление видимостью кнопки "Назад" в приложении

Давайте сделаем пользовательскую кнопку возврата видимой, только когда в стеке переходов назад есть что-то.

1. Добавьте закрытое поле **Visibility (Видимость)** в выделенный код Страницы 2.

C#

```
public sealed partial class Page2: Page
{
    public Visibility CanGoBack;
```

2. В переопределении страницы 2 **OnNavigatedTo** проверьте, может ли фрейм быть возвращён, и настройте поле видимости соответственно.

C#

```
protected async override void OnNavigatedTo(NavigationEventArgs e)
{
    if (Frame.CanGoBack)
        CanGoBack = Visiblity.Visible;
    else
        CanGoBack = Visibility.Collapsed;
```

```
    await new MessageDialog("You sent: " + e.Parameter).ShowAsync();
}
```

3. Свяжите видимость своей кнопки возврата с полем **CanGoBack** на **Странице 2.xaml**.

XAML

```
<Button Style="{StaticResource NavigationBackButtonNormalStyle}"
        Click="{x:Bind GoBack}" Visibility="{x:Bind CanGoBack}"
        VerticalAlignment="Top" />
```

4. В **App.xaml.cs** настройте первоначальную страницу запуска для своего приложения, чтобы она указывала на **Page2** вместо **MainPage**.

C#

```
if (rootFrame.Content == null)
{
    // Когда стек навигации не может быть восстановлен, перемещайтесь к
первой странице,
    // конфигурируя новую страницу, передавая требуемую информацию
    // как параметр навигации
    rootFrame.Navigate(typeof(Page2)), e.Arguments);
}
```

5. Скомпилируйте и запустите своё приложение. Пользовательская кнопка "Назад" не появится на странице 2, потому что в стек переходов назад пуст.

Рисунок 13

Пользовательская кнопка "Назад" не появляется, когда стек переходов назад пуст.

6. Отключите отладку и вернитесь к **App.xaml.cs**. Замените страницу запуска обратно на **MainPage**.

C#

```
if (rootFrame.Content == null)
{
    // Когда стек навигации не может быть восстановлен, перемещайтесь к
    // первой странице, конфигурируя новую страницу, передавая требуемую
    // информацию как параметр навигации
    rootFrame.Navigate(typeof(MainPage)), e.Arguments);
}
```

7. Скомпилируйте и запустите своё приложение. Перейдите на страницу 2. Пользовательская кнопка "Назад" снова появится, когда стек будет заполнен.

8. Завершите отладку и вернитесь к Visual Studio.

Краткий обзор

В этой работе вы узнали о фреймах, навигации между страницами и поведении при переходе назад в пределах приложения UWP. Хотя опции навигации варьируются для разных устройств, вы научились создавать приложения, которые одинаково хорошо ведут себя на на настольном компьютере, планшете и мобильном устройстве, подстраиваясь под принятый для данного форм-фактора стиль навигации.