Разработка приложений для Windows Phone 7

Это новая мобильная платформа, которая позволяет создавать интерактивные приложения и игры на Silverlight и XNA для пользователей по всему миру, объединенных сетевыми сервисами.

Учебный курс от новичка к эксперту

- Первое знакомство с платформой
- Шаблон проекта, страницы и навигация
- Стандартные варианты разметки
- Элементы управления Pivot и Panorama
- Основные элементы управления
- Текстовые поля и контекст ввода
- Задачи запуска (Launchers)
- Задачи выбора (Choosers)
- Элемент управления Map
- Элемент управления WebBrowser
- Работа с акселерометром
- Определение местоположения
- Работа с HTTP
- Работа с изолированным хранилищем
- Знакомство с локальной базой данных
- Жизненный цикл и сохранение состояния приложения
- Фоновые сервисы, запускаемые по расписанию
- Фоновая загрузка/выгрузка файлов
- Фоновое проигрывание музыки
- Оповещения
- Живые тайлы
- Push Notification

Новости

Visual Studio 11 и разработка приложений для Windows 8 вошли в число ключевых тем на конференции DevCon'12
2 марта в Москве прошла конференция Windows 8 Camp, во время которой разработчики познакомились с особенностями разработки Metro-приложений для Windows 8 с помощью Visual Studio 11, beta-версия которо... подробнее
понедельник, мар 5 **I.Vorontsov**

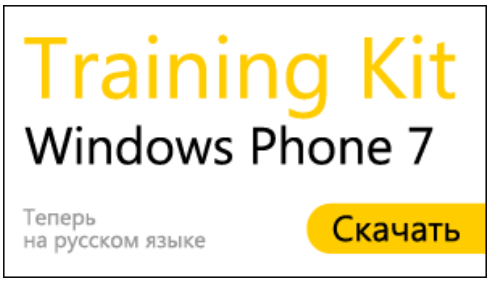
Новые материалы на Русском MSDN!
Уважаемые коллеги!Ниже приведена подборка подготовленных и опубликованных на MSDN материалов за последнии 10 дней:WebРепозиторий Helicon Zoo. Знакомство с GoliathWindows PhoneВсё о Splash Scre... подробнее
вторник, фев 28 **I.Vorontsov**

Опубликовано назнавние первых 100 докладов конференцииTechEd Europe 2012
Опубликовано назнавние первых 100 докладов крупнейшей европейской конференции TechEd Europe 2012, которая пройдет с 26 по 29 июня 2012 года в Амстердаме . В ходе конференции посетители смогут посетить... подробнее
вторник, фев 21 **I.Vorontsov**

Запущена Beta версия нового крупнейшего технологического видео хостинга TechDays!
Запущена beta версия нового технологического видео хостинга techdays! Если вы специалист в области разработки программного обеспечения или администрирования ИТ систем данный ресурс именно для вас. ... подробнее
понедельник, фев 20 **I.Vorontsov**

Все новые материалы Русского MSDN в одном агрегаторе
На Русском MSDN появилась возможность подписаться на RSS со всеми новыми материалами ресурса. Так же можно подписаться на RSS только по интересующему вас направлению в разработке, на данный момент дос... подробнее
четверг, фев 16 **I.Vorontsov**

Видео-доклады по Windows Phone 7 на русском языке



Новые материалы

Всё о Splash Screen в Windows Phone

Метро-дизайн. Навигация, уровни, назад и циклы

Метро-дизайн. Темы и акценты

Метро-дизайн. Контрастность и доступность

Метро-дизайн. Практика. Градиенты и 16-битность

Архив обновлений

Полезные ресурсы

- Хаб по Metro-дизайну
- Разработка 3D игр
- Центральный ресурс для разработчиков под Windows Phone 7
- Руководство по публикации приложений в Marketplace
- Руководство по публикации приложений в Marketplace для студентов
- Группа разработчиков под Windows Phone 7 на Facebook
- Мастер-класс по дизайну приложений для Windows Phone
- Дизайн приложений для WP7. Metro-подход
- Серия статей "45 дней с Windows Phone 7
- Узнайте больше о Windows Phone 7
- **Дополнительные материалы**

Блог

Промежуточные итоги конкурса - 23 февраля
В конце января мы совместно с Nokia запустили конкурс весенних приложений для Windows Phone. Конкурс... дополнительно
пятница, фев 24 **Mik Chernomordikov**

Промежуточные итоги конкурса - 14 февраля
В конце января мы совместно с Nokia запустили конкурс весенних приложений для Windows Phone. Конкурс... дополнительно
среда, фев 15 **Mik Chernomordikov**

Литература на зиму - новый WP7 Training Kit
Длинные новогодние выходные – хорошая возможность засесть за изучение технологий! Тем более, что не... дополнительно
вторник, янв 3 **Mik Chernomordikov**

Праздник в Marketplace приходит - результаты!
Месяц назад мы анонсировали совместный с Nokia конкурс новогодних приложений – и вот пришло вр... дополнительно
понедельник, дек 26 **Mik Chernomordikov**

Форум

Использование зарегистрированных торговых марок в приложении.
Можно ли использовать логотипы типа Adidas или KFC и т.п. в приложении? Если можно то есть ли ... дополнительно
понедельник, мар 12

Задай свой вопрос MVP - эксперту по Мобильной разработке (WP7 и XNA)
Здравствуйте.В данной теме на ваши вопросы будут отвечать эксперты в мобильной разработ... дополнительно
воскресенье, мар 11

windows mobile 6.5 & visual studio 2010 ultimate
Здравствуйте! У меня вот такой вопрос! Как создавать приложения под windows mobile 6.5 на C#? ... дополнительно
суббота, мар 10

Стилизации Button
День добрый.Стоит следующая задача. Нужно стилизовать кнопку. Есть две картинки, соответственно для... дополнительно
четверг, мар 8

Рекомендуемые книги

Программируем Windows Phone 7. (Полное издание)

Разработка под Windows Phone: Часть 1: Инструментарий разработки, шаблоны и первое приложение

Рейтинг

Прежде чем приступить к знакомству с возможностями платформы и разработке приложений, необходимо убедиться, что у нас есть весь необходимый инструментарий и разобраться с некоторыми основами разработки.

Инструментарий

Visual Studio 2010

Чтобы разрабатывать для Windows Phone потребуется Visual Studio 2010 с Service Pack 1 редакции Professional или выше. Если у вас нет Visual Studio 2010, при установке инструментарий разработки для Windows Phone, автоматически будет установлена бесплатная версия Visual Studio 2010 Express for Windows Phone, на которой также можно разрабатывать приложения под Windows Phone.

Обе версии интегрированных средств разработки Visual Studio предоставляют разработчику полноценные возможности по отладке на устройстве и эмуляторе такие же, какие есть у разработчиков приложений под настольную версию Windows.

Обратите внимание, что для того, чтобы отлаживаться на устройстве, помимо собственно устройства и кабеля для подключения его к компьютеру разработчика, на компьютере со средствами разработки необходимо иметь установленное ПО Zune (http://zune.net). Также перед развертыванием приложения и отладкой, требуется зарегистрировать устройство или «разлочить», с использованием утилиты Windows Phone Developer Registration Tool, которая устанавливается вместе с Windows Phone SDK.

Windows Phone SDK

Этот пакет, доступный для скачивания на сайте App Hub http://create.msdn.com содержит всё необходимое, для того, чтобы начать разработку. На момент написания этой статьи, последняя версия инструментария доступна в версии Windows Phone SDK 7.1 Release Candidate в лицензии «Go Live» с возможностью разрабатывать свои приложения и публиковать их в Windows Phone Marketplace. Windows Phone SDK 7.1 Release Candidate содержит следующие компоненты:

- Windows Phone SDK 7.1
- Windows Phone Emulator
- Windows Phone SDK 7.1 Assemblies
- Silverlight 4 SDK and DRT
- Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0
- Expression Blend SDK for Windows Phone 7
- Expression Blend SDK for Windows Phone OS 7.1
- WCF Data Services Client for Windows Phone
- Microsoft Advertising SDK for Windows Phone

Если у вас не установлена версия Visual Studio 2010 редакции Professional, Expression Bland 4 или XNA Game Studio 4.0, в процессе установки также будут скачаны и установлены:

- Visual Studio 2010 Express for Windows Phone
- Expression Blend 4 for Windows Phone
- XNA Game Studio 4.0

Expression Blend и Expression Blend for Windows Phone

Expression Blend – это интерактивный визуальный дизайнер для XAML, технологии описания интерфейса для приложений Silverlight и Windows Presentation Foundation (WPF). Это отличное средство разработки, которое позволяет просто манипулировать слоями, анимацией, стилями и шаблонами. Это базовое средство разработки на XAML. Собственно программа Expression Blend не бесплатна, однако, специальная версия для создания дизайнов приложений под Windows Phone, под названием Expression Blend 4 for Windows Phone доступна для разработчиков бесплатно. Она закачается и установится в процессе установки Windows Phone SDK, если у вас на компьютере нет полной версии Expression Blend. Подробнее об Expression Blend 4 можно прочитать на MSDN: http://msdn.microsoft.com/ru-ru/library/cc296227.aspx

XNA Game Studio 4.0

От новичка к эксперту

XNA Game Studio – это программное окружение, которое позволяет разрабатывать в Visual Studio игры для Windows Phone, консоли Xbox 360 и компьютеров на базе Windows. Включает в себя XNA Framework, представляющий собой набор библиотек на управляемом коде для разработки игр. Подробнее можно прочитать на MSDN: http://msdn.microsoft.com/ru-ru/library/bb200104.aspx

Windows Phone Emulator

Несмотря на то, что Windows Phone Emulator не содержит полного набора приложений доступных на реальном устройстве, он предоставляет мощную среду, позволяющую практически полностью разработать приложение в эмуляторе.

Эмулятор Windows Phone Emulator не поддерживает проигрывание медиаконтента Zune. Эмулятор поставляется только с одним встроенным приложением Internet Explorer, но это Internet Explorer 9 с поддержкой HTML5.

При этом эмулятор позволяет тестировать звонки и отсылку SMS сообщений, поддерживает мультитач на мониторах с поддержкой мультитач, поддерживает симуляцию камеры, геолокационных сервисов и акселерометра, а также позволят делать снимки экрана.

Подробнее можно прочитать на MSDN: http://msdn.microsoft.com/ru-ru/library/ff402563(v=VS.92).aspx

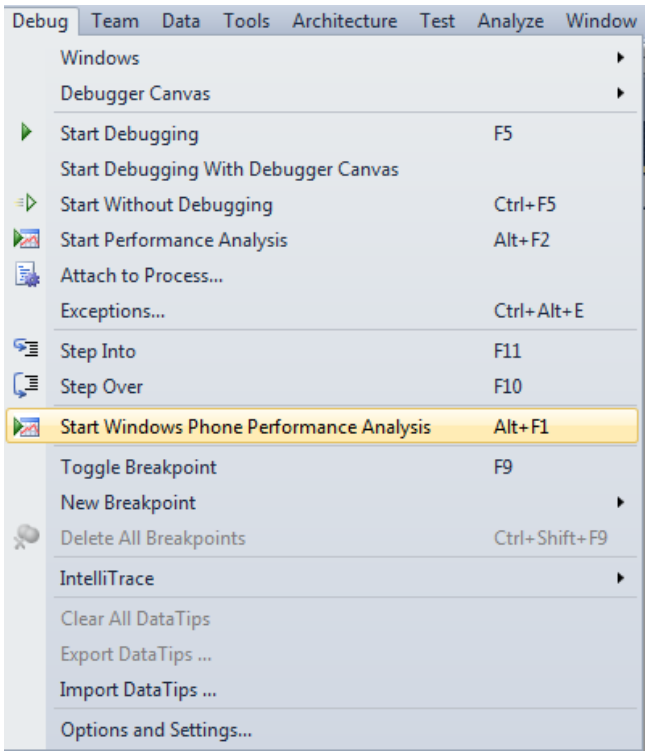
Дополнительный инструментарий разработчика

Windows Phone Developer Registration Tool

Перед тем, как разработчик сможет развернуть своё приложение на реальном устройстве, его необходимо зарегистрировать как устройство разработчика – «разлочить». Это делается один раз для определенного телефона. Зарегистрированный на Marketplace разработчик может зарегистрировать до 3 устройств (для разработчика, зарегистрированного, как студент количество устройств ограничено до одного). Подробнее: http://create.msdn.com

Windows Phone Profiler

Windows Phone Profiler доступен в меню Debug Visual Studio с установленным инструментарием Windows Phone SDK.



Анализирует работу программы во время исполнения , идентифицирует возможные проблемы с производительностью. Подробнее можно прочитать на MSDN: http://msdn.microsoft.com/ru-ru/library/hh202934(v=VS.92).aspx

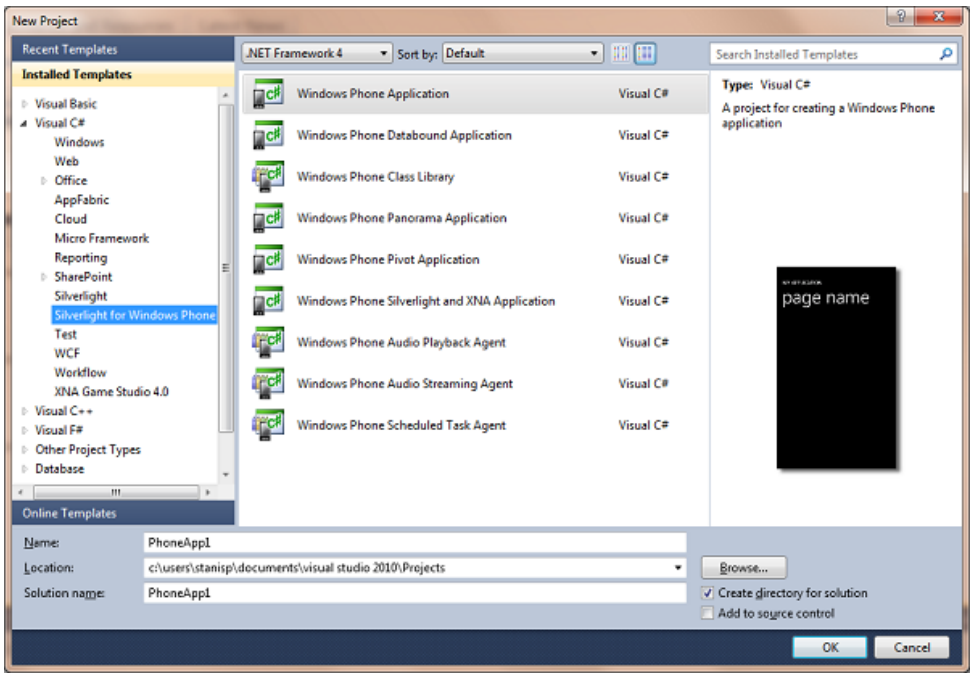
Silverlight Toolkit for Windows Phone

Silverlight Toolkit for Windows Phone – набор полезных элементов управления Silverlight дляWindows Phone с поддержкой режима дизайна, от команды разработчиков Silverlight. Доступен весь исходный код, примеры и документация. Обновляется приблизительно раз в три месяца, доступен по адресу http://silverlight.codeplex.com или через NuGet.

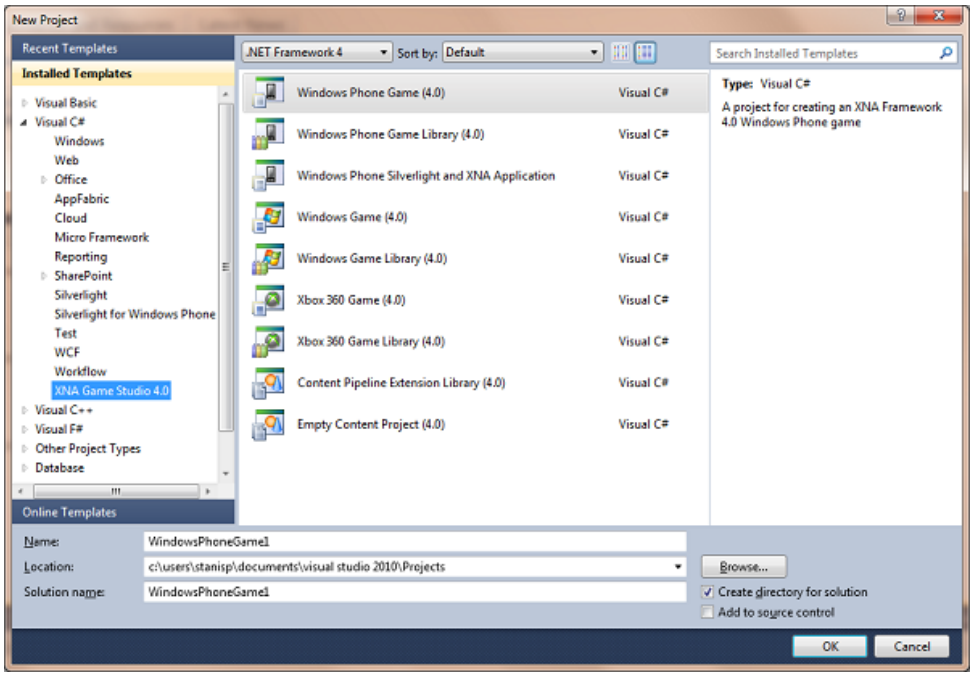
Текущий релиз включает в себя такие элементы управления, как ContextMenu, DatePicker и TimePicker, ToggleSwitch, WrapPanel и GestureHelper.

Среда разработки

После установки средств разработки Windows Phone SDK в диалоге New Project в Visual Studio появятся группы проектов для Silverlight for Windows Phone:



и в группе XNA Game Studio 4.0 добавятся проекты для Windows Phone:



Этот цикл статьей сфокусирован на разработке под Windows Phone на Silverlight, поэтому рассмотрим доступные разработчику приложений шаблоны несколько более подробно.

После установки разработчику доступны следующие шаблоны приложений Silverlight for Windows Phone:

- Windows Phone Application
- Windows Phone Databound Application
- Windows Phone Class Library
- Windows Phone Panorama Application
- Windows Phone Pivot Application
- Windows Phone Silverlight and XNA Application
- Windows Phone Audio Playback Agent
- Windows Phone Audio Streaming Agent

- Windows Phone Scheduled Task Agent

Перед тем как перейти непосредственно к шаблонам приложений, надо сказать несколько слов по поводу Windows Phone и Metro-дизайна.

Windows Phone и Metro-дизайн

Платформа Windows Phone не просто очередная платформа для мобильных устройств. Она содержит в себе не только технологическую составляющую, но и полностью проработанную концепцию дизайна интерфейса и взаимодействия с пользователем под названием Metro-дизайн или стиль Metro.

Если вы дизайнер или в вашей команде есть выделенный дизайнер, вы можете воспользоваться всей мощью инструментарий Expression Blend 4 или Expression Blend for Windows Phone, которая поставляется вместе с Windows Phone SDK.

Что же делать если вы разработчик и не хотите заниматься визуальным дизайном приложения, например, вы разрабатываете бизнес-приложение и всё что от него требуется, соответствовать общему дизайну и стилю Windows Phone?

Всё очень просто. Во-первых, Silverlight для телефона разработан с учётом требований Metro-дизайна, поэтому все встроенные элементы управления выполнены в Metro-дизайне. Во-вторых, по умолчанию, приложения, созданные из шаблонов из поставки Windows Phone SDK, работают, выглядят и используют стили и шрифты в соответствии с Metro-дизайном.

С другой стороны, возможностей стилизации элементов управления и приложений, основанных на XAML, которые представляет Silverlight, вполне достаточно, чтобы сделать своё приложение неповторимым и узнаваемым, оставаясь в рамках стиля Metro.

Руководство по дизайну интерфейсов и взаимодействию с пользователем для Windows Phone можно найти по следующей ссылке http://msdn.microsoft.com/ru-ru/library/hh202915.aspx

Всё что было сказано выше относится, конечно, к дизайну обычных приложений, так как требования к дизайну игровых приложений и их интерфейсу, могут существенно отличаться. При этом не надо забывать об общих принципах взаимодействия с пользователем, заложенных в концепции Windows Phone.

Шаблоны приложений

Сначала давайте рассмотрим три шаблона, представляющих собой три основных стиля приложения для Windows Phone:

- Windows Phone Application
- Windows Phone Pivot Application
- Windows Phone Panorama Application

Windows Phone Application – это аналог простого диалогового приложения у которого один основной экран, через который происходит основное взаимодействие с пользователем.

Windows Phone Pivot Application – это некий аналог приложения с закладками, где заголовок каждой закладки определяет содержимое. Стандартный вариант использования – каждая закладка представляет собой одни и те же, в целом, данные, но в разных представлениях и/или с разной фильтрацией. Например, календарь, почтовый клиент и настройки телефона. Шаблон использует элемент управления Pivot.

Windows Phone Panorama Application – приложение панорама, в котором зоны взаимодействия с пользователем также разделены на панели, но доступны они через горизонтальную прокрутку; фоновое изображение установлено сразу на всю панораму, она имеет общий заголовок, который прокручивается медленнее, чем панели; контент соседней панели справа виден при отображении

текущей. Например, таким образом реализованы хабы в Windows Phone: People, Marketplace, Pictures, Music+Videos и др. Шаблон использует элемент управления Panorama.

Шаблоны, заканчивающиеся на Agent – это шаблоны библиотек, для выполнения соответствующих фоновых задач:

- Windows Phone Audio Playback Agent
- Windows Phone Audio Streaming Agent
- Windows Phone Scheduled Task Agent

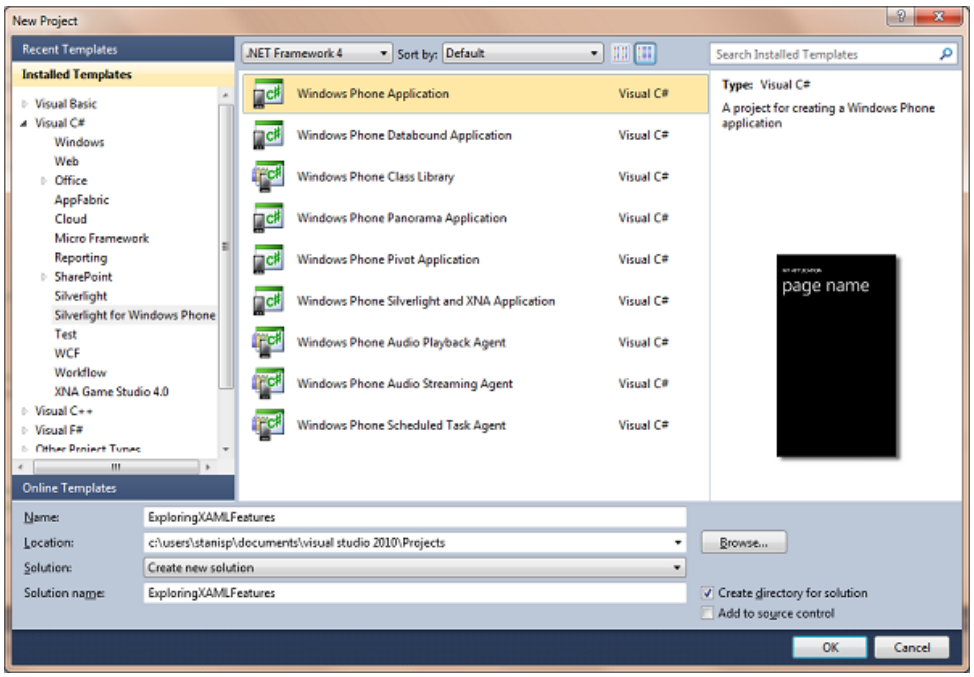
Шаблон Windows Phone Databound Application – простой шаблон приложения с вида список – детальное представление с реализацией навигации между страницами с передачей параметров и хранением данных в глобальном VeiwModel.

Шаблон Windows Phone Class Library – шаблон библиотеки классов для Windows Phone.

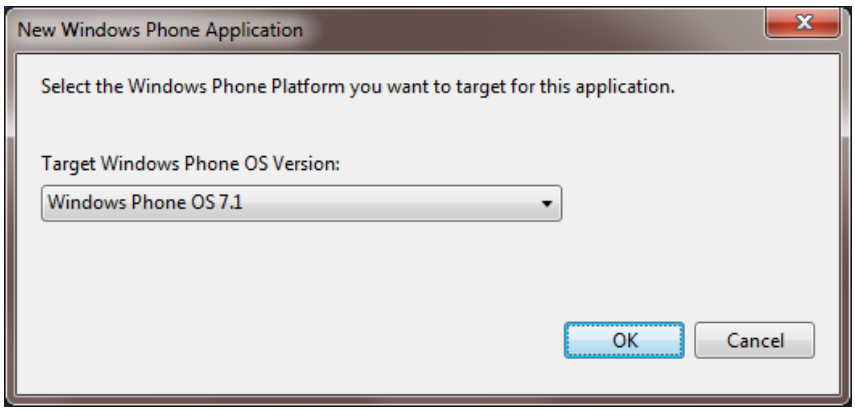
Шаблон Windows Phone Silverlight and XNA Application для Silverlight приложения, которое может использовать XNA для рендеринга графического контента.

Создание простого приложения

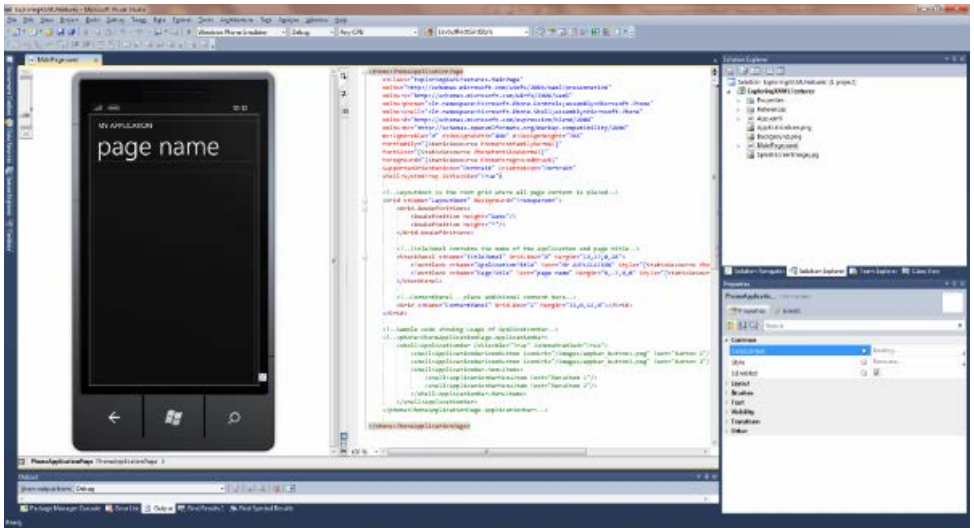
В диалоге New Project Visual Studio выберем Visual C#, Silverlight for Windows Phone и простой шаблон приложения Windows Phone Application и назовём его ExploringXAMLFeatures.



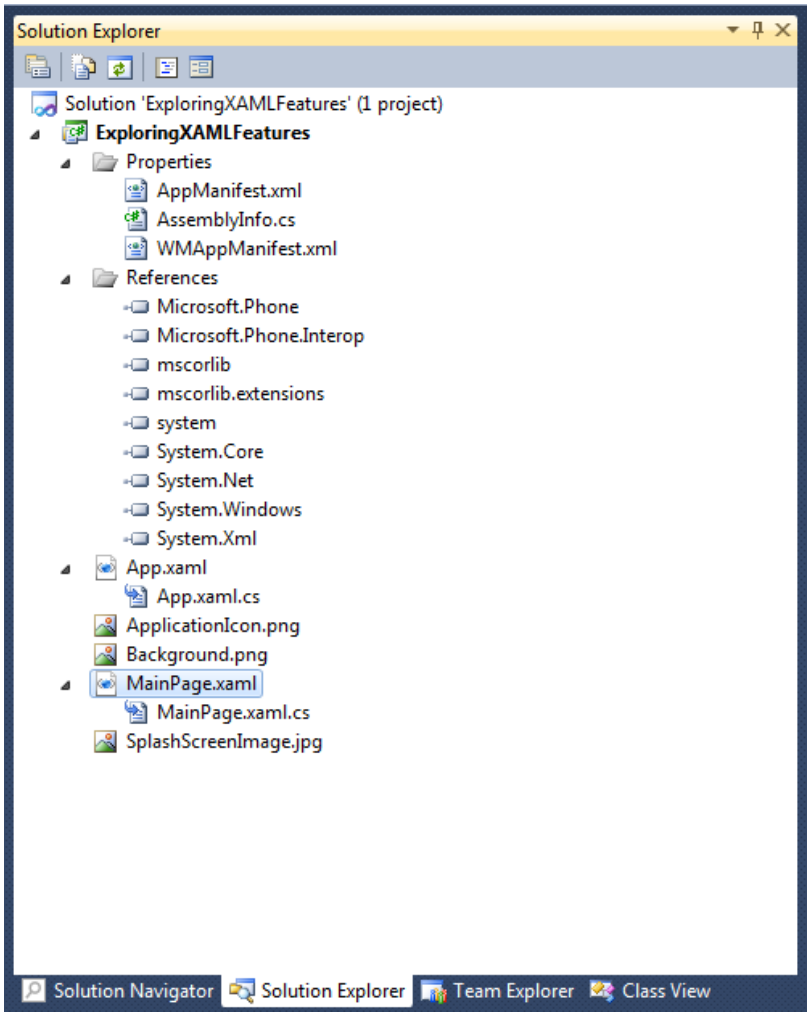
В диалоге выбора целевой операционной системы выберем Windows Phone OS 7.1



После создания проекта, окно Visual Studio примет следующий вид



Рассмотрим структуру проекта в окне Solution Explorer:



Название файла	Назначение
AppManifest.xml	Файл манифеста, необходимый для генерации XAP файла, в который упаковывается приложение для развёртывания на телефоне.
AssemblyInfo.cs	Ещё один конфигурационный файл, в котором определяются некоторые метаданные главной сборки(Assembly) приложения.
WMAppManifest.xml	Файл метаданных, который содержит разнообразные настройки приложения: заголовок, задание первой страницы, пути к иконкам, определение необходимых системных возможностей и.т.д.
App.xaml	Это файл ресурсов приложения. Здесь располагаются глобальные ресурсы

	(это будет рассмотрено при использовании стилей) или глобальные события (происходящие при старте приложения). Этот файл также является точкой входа приложения.
App.xaml.cs	Файл кода (code-behind) для App.xaml. Здесь можно обрабатывать события и ошибки уровня приложения, в том числе его tombstoning. Данную концепция будет рассмотрена позднее, когда будет рассматриваться многозадачность.
ApplicationIcon.png	Картинка, которая будет иконкой приложения в телефоне. Это действительно важный файл, так как он является первым, что увидят пользователи при работе с приложением.
Background.png	Данная картинка используется, когда приложение закреплено на стартовом экране телефона(start screen). По сути это большая иконка приложения. Разумно сделать её визуально похожей на ApplicationIcon.png.
MainPage.xaml	Это часть выбранного шаблона приложения. Название MainPaige не очень удачное, но именно оно используется шаблоном проекта по умолчанию. Этот страница представляет интерфейс, который видит пользователь при старте приложения.
MainPage.xaml.cs	Файл кода страницы MainPage.xaml.
SplashScreenImage.jpg	Данная картинка отображается во время загрузки +приложения. Можно задать свою картинку с анимацией, чтобы проинформировать, что приложение загружается. Есть техника создания очень динамичных страниц загрузки на XNA, но она выходит далеко за рамки этого цикла статей.

Файлы XAML определяют интерфейс приложения. На самом деле - это просто XML файлы с языком разметки XAML.

Несмотря, что это самый простой проект, он содержит все ключевые элементы, которые содержат все остальные шаблоны и типы проектов.

Обратите внимание, что часть настроек, представленных в виде конфигурационных файлов, могут редактироваться в визуальном интерфейсе редактирования настроек приложения.

Добавление элементов управления на страницу XAML

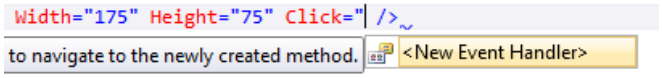
Обратите внимание, что Visual Studio по умолчанию отображает и дизайн, и XAML код страницы.

Если вы переходили к просмотру других файлов решения, двойным щелчком перейдите на файл MainPage.xaml.

В XAML код файла MainPage.xaml внутрь элеметна Grid с именем ContentPanel вставьте элемент управления Button:

```
1. <!--ContentPanel - place additional content here-->
2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
3.  <Button Content="Нажми меня" Name="MyButton" FontSize="18" Width="175" Height="75" />
4. </Grid>
```

В окне дизайна кнопка отобразится сразу приблизительно в центре интерфейса. Обратили внимание на атрибут **Name**? Это уникальный идентификатор элемента, который помогает ссылаться на него в коде. Считайте это ID атрибутом элемента управления. Давайте теперь добавим какие-нибудь действия при нажатии на эту кнопку. Есть два способа привязать событие к кнопке Button (или любому другому элементу управления). В XAML, прямо в определении Button, можно добавить атрибут Click и система InteliSense автоматически спросит, хотим ли мы сгенерировать новый обработчик событий:



Можно связать обработчик событий напрямую в коде страницы Home.xaml.cs не указывая его в XAML файле:

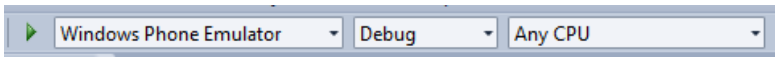
```
1. public MainPage()
2. {
3.      InitializeComponent();
4.      MyButton.Click +=new RoutedEventHandler(MyButton_Click);
5. }
```

Оба способа работают. Можете использовать любой из них. Для упрощения здесь будет использоваться определение метода в XAML. Теперь в функции MyButton_Click можно написать

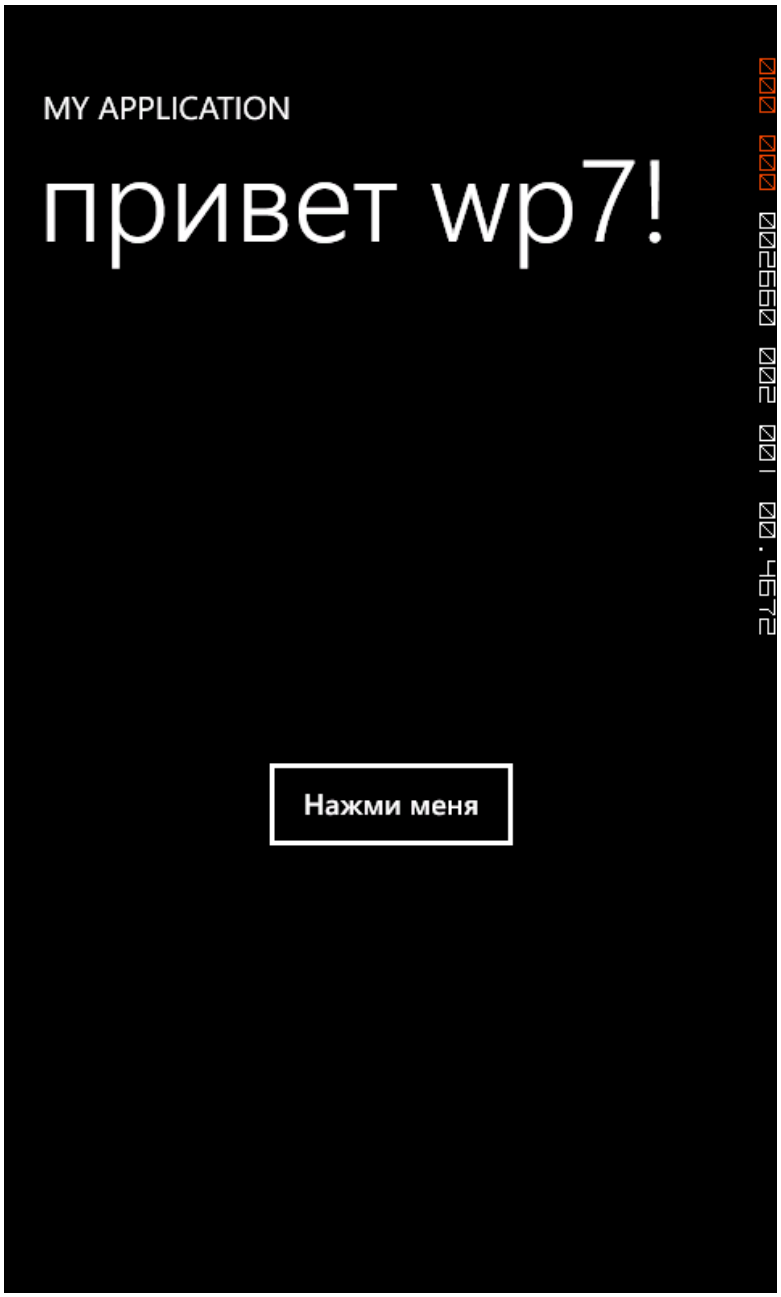
управляемый код, который будет изменять интерфейс или вызывать другие функции. Завершим наш пример приложения, добавив код, который будет изменять текст в TextBlock PageTitle (PageTitle – это Name, так что можно ссылаться на неё напрямую в коде) на «привет wp7». Чтобы сделать это допишем следующий код в функцию:

```
1. private void MyButton_Click(object sender, RoutedEventArgs e)
2. {
3.         PageTitle.Text = "привет wp7";
4. }
```

Выберем в настройках проекта Windows Device Emulator



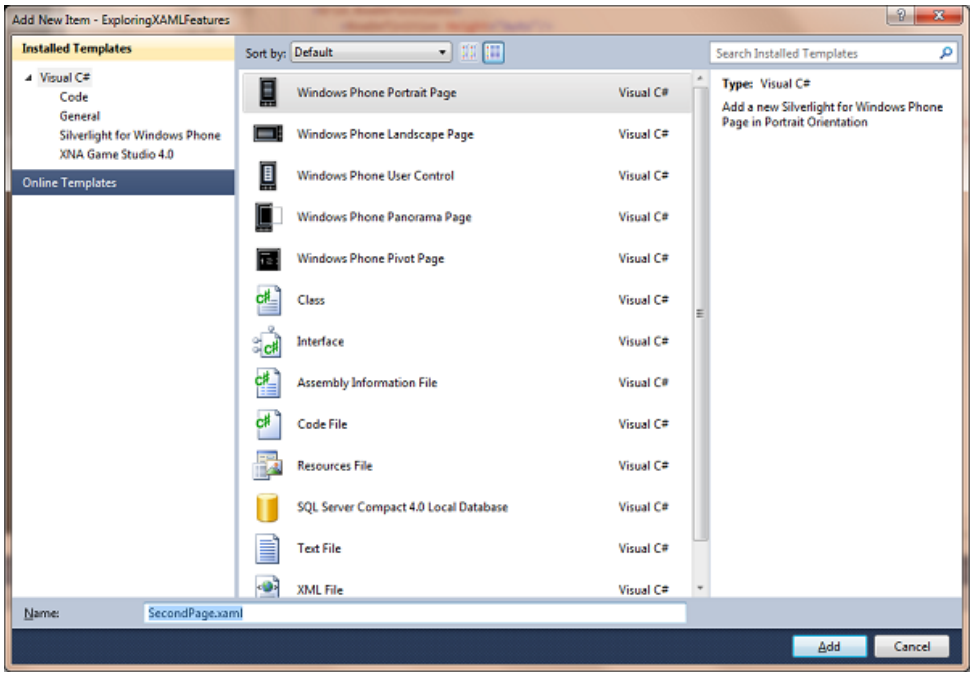
И запустим приложение, нажав на зеленый треугольник или копку F5. После запуска приложения и нажатия на кнопку «Нажми меня», вид экрана должен быть аналогичным снимку экрана ниже:



Добавление новых страниц в проект

Только самое простое приложение состоит из одной страницы. Мы хотим научиться писать сложные многостраничные приложения. Мы можем использовать шаблоны Pivot, Panorama, можем использовать паттерн проектирования MVVM (Model-View-ViewModel), а сначала научимся добавлять новые страницы в проект и переходить между ними.

В окне Solution Explorer щелкнем правой кнопкой мыши по названия проекта, и в отобразившемся меню выберем, Add, далее New Item, в открывшемся диалоговом окне выберем Windows Phone Portrait Page и назовем её SecondPage.xaml:



Теперь у нас есть пустая XAML страница, точная копия страницы MainPage.xaml до того, как мы её отредактировали.

Чтобы лучше различать страницы, перейдем к XAML коду страницы SecondPage и у элемента TextBlock с Name PageTitle отредактируем свойство Text, как показано ниже:

```
<TextBlock Name="PageTitle" Text="second page" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
```

Навигация между страницами приложения

Итак, у нас в проекте есть две страницы, при запуске приложения отображается страница MainPage.xaml. Как теперь перейти со страницы MainPage.xaml на SecondPage.xaml?

Попробуем два простых способа, как это сделать.

В XAML код файла MainPage.xaml после добавленного ранее кода Button, добавим код HyperlinkButton, как показано ниже:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <Button Content="Нажми меня" Name="MyButton" FontSize="18" Width="175" Height="75" />
        <HyperlinkButton Content="Вторая страница" Name="MyHyperlinkButton" FontSize="18" Wid
th="175" Height="75" Margin="140,350,140,185" />
</Grid>
```

Элемент управления HyperlinkButton имеет специальное свойство NavigateUri, которое позволяет указать Uri для перехода. Добавим это свойство со значением /SecondPage.xaml, как показано ниже:

```
<HyperlinkButton Content="Вторая страница" Name="MyHyperlinkButton" FontSize="18" Width="175"
 Height="75" Margin="140,350,140,185" NavigateUri="/SecondPage.xaml"/>
```

Запустим приложение (F5).

Когда отобразится интерфейс приложения, если мы нажмём ссылку Вторая страница, то произойдет переход на вторую, ранее созданную страницу SecondPage.xaml. Если после этого нажать на аппаратную кнопку Back, то мы вернёмся на основную (предыдущую) страницу – по умолчанию эта кнопка позволяет перейти на предыдущую активную страницу.

Теперь воспользуемся возможностью программного перехода. Сначала добавим в секцию using следующий код:

```
using System.Windows.Navigation;
```

А затем, заменим код обработчика MyButton_Click на следующий:

```
private void MyButton_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Uri("/SecondPage.xaml", UriKind.Relative));
}
```

Запустите приложение (F5) и убедитесь, что кнопка работает также как и ссылка.

Добавим на вторую страницу (SecondPage.xaml) элементы управления и код, который бы позволял вернуться на предыдущую страницу.

В XAML код файла SecondPage.xaml внутрь элеметна Grid с именем ContentPanel вставьте элементы управления Button и HyperlinkButton, как показано ниже:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <Button Content="Обратно" Name="MyButton" FontSize="18" Width="175" Height="75" Click="MyBu
tton_Click" />
  <HyperlinkButton Content="Первая страница" Name="MyHyperlinkButton" FontSize="18" Width="17
5" Height="75" Margin="140,350,140,185" NavigateUri="/MainPage.xaml"/>
</Grid>
```

Для создания обработчика события Click кнопки MyButton на странице SecondPage кликните правой кнопкой мыши по тексту MyButton_Click в XAML редакторе и выберите Navigae to Event Handler.

Запустите приложение (F5) и убедитесь, что ссылка «Первая страница» на второй странице возвращает приложение на MainPage.

Теперь воспользуемся возможностью программного перехода. В коде страницы SecondPage.xaml.cs добавим в секцию using следующий код:

```
using System.Windows.Navigation;
```

А затем, заменим код обработчика MyButton_Click на следующий:

```
private void MyButton_Click(object sender, RoutedEventArgs e)
{
   NavigationService.GoBack();
}
```

Запустите приложение (F5) и убедитесь, что кнопка Обратно на второй странице возвращает приложение на MainPage.

Обратите внимание, что в коде, который запускается по нажатию кнопки мы не использовали название страницы, на которую нам необходимо перейти, а попросили сервис навигации, представленный классом NavigationService перейти на страницу, которая была в стеке переходов перед текущей.

События перехода с/на страницы могут быть обработаны програмно, например, чтобы уточнить у пользователя, действительно ли он хочет уйти с текущей страницы.

Добавьте следующий код в файл SecondPage.xaml.cs сразу после обработчика MyButton_Click:

```
protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
   base.OnNavigatingFrom(e);

   // Если можно отменить переход, уточним у пользователя, хочет ли он остаться на текущей с
транице
   if (e.IsCancelable)
   {
     MessageBoxResult result = MessageBox.Show("Может быть останетесь?
", "Подтверждение перехода", MessageBoxButton.OKCancel);
        if (result == MessageBoxResult.OK)
         {
            // Пользователь решил остаться
            e.Cancel = true;
            return;
          }
      }
```

```
16.  }
```

Код достаточно простой, чтобы вы с ним могли разобраться самостоятельно.

Запустите приложение (F5) и обратите внимание, что уточняющий диалог отображается вне зависимости от того, собираемся вы вернуться на MainPage нажатием на кнопку или по ссылке.

Теперь осталось научиться передавать между страницами параметры.

Двойным щелчком по MainPage.xaml перейдём к редактированию основной страницы. В коде MainPage.xaml добавим элемент управления TextBox, выше элемента Button, как показано ниже:

```
1. <!--ContentPanel - place additional content here-->
2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
3.  <TextBox Name="MyTextBox" Width="175" Height="75" Margin="140,185,140,350"/>
4.  <Button Content="Нажми меня" Name="MyButton" FontSize="18" Width="175" Height="75" Click="My
   Button_Click"/>
5.  <HyperlinkButton Content="Вторая страница" Name="MyHyperlinkButton" FontSize="18" Width="175
   " Height="75" Margin="140,350,140,185" NavigateUri="/SecondPage.xaml"/>
6. </Grid>
```

Двойным щелчком по SecondPage.xaml перейдём к редактированию основной страницы. В коде SecondPage.xaml добавим элемент управления TextBox, выше элемента Button, также как на странице MainPage:

```
1. <!--ContentPanel - place additional content here-->
2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
3.    <TextBox Name="MyTextBox" Width="175" Height="75" Margin="140,185,140,350"/>
4.    <Button Content="Обратно" Name="MyButton" FontSize="18" Width="175" Height="75" Click="MyB
   utton_Click" />
5.    <HyperlinkButton Content="Первая страница" Name="MyHyperlinkButton" FontSize="18" Width="1
   75" Height="75" Margin="140,350,140,185" NavigateUri="/MainPage.xaml"/>
6. </Grid>
```

Теперь, в обработчик события MyButton_Click страницы MainPage добавим параметры в Uri перехода на вторую страницу по кнопке:

```
1. private void MyButton_Click(object sender, RoutedEventArgs e)
2. {
3.     NavigationService.Navigate(new Uri("/SecondPage.xaml?
   text="+Uri.EscapeDataString(MyTextBox.Text), UriKind.Relative));
4. }
```

И, наконец, в код страницы SecondPage.xaml.cs, сразу же после обработчика OnNavigatedFrom, добавим обработчик прихода на страницу OnNavigatedTo и в нем обработаем приходящий параметр и выведем его в MyTextBox:

```
1. protected override void OnNavigatedTo(NavigationEventArgs e)
2. {
3.    base.OnNavigatedTo(e);
4.
5.    if (NavigationContext.QueryString.ContainsKey("text"))
6.    {
7.       MyTextBox.Text = NavigationContext.QueryString["text"].ToString();
8.    }
9.
10. }
```

Запустите приложение (F5) и проверьте, как оно работает, например, при переходе по ссылке со страницы MainPage на SecondPage.

Итоги и следующие шаги

Итак, мы разобрались с тем, что надо установить для разработки под Windows Phone 7, познакомились с доступными шаблонами и научились создавать новое приложение из шаблона. Также мы научились добавлять элементы управления, прямо в XAML код, добавлять в проект новые страницы и организовывать между ними переход по ссылке и в коде с возможностью передачи параметров.

На следующем шаге мы познакомимся с некоторыми вариантами разметки, доступными в Silverlight, рассмотрим основные элементы управления, а также узнаем, что такое контекст ввода для текстовых полей.

Файлы для загрузки

Проект ExploringXAMLFeatures

Разработка под Windows Phone: Часть 2: Варианты разметки, основные элементы управления и контекст ввода

Рейтинг

В предыдущей части мы научились добавлять элементы управления, используя код XAML. В этой части мы познакомимся с некоторыми основными доступными разработчику вариантами разметки, посмотрим, какие элементы нам доступны в Toolbox и узнаем, что такое контекст ввода текстовых полей

Стандартные варианты разметки

Silverlight предоставляет гибкую систему для размещения элементов интерфейса страницы приложения. Есть модели разметки, которые поддерживают и динамические и абсолютные стили разметки. Есть достаточно много элементов управления, предоставляющих управление разметкой в Silverlight, самые используемые, это:

- Canvas
- StackPanel
- Grid

Давайте рассмотрим, как каждый из элементов работает, разметив внутри них другие элементы управления. В целях демонстрации, мы будем использовать простые элементы управления Button. Для этого создайте новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и назовите его ExploreBaseControls. Обратите внимание, что весь код, для простоты, будем вставлять в страницу MainPage.xaml внутрь элемента Grid:

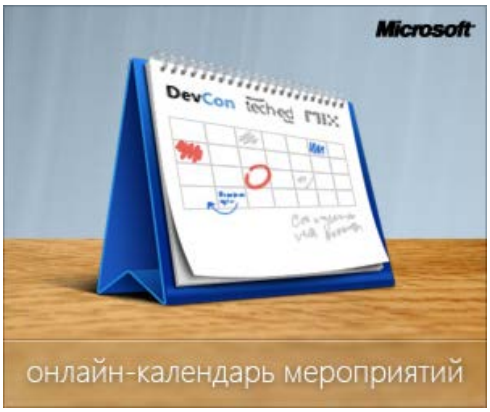
```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
```

Canvas

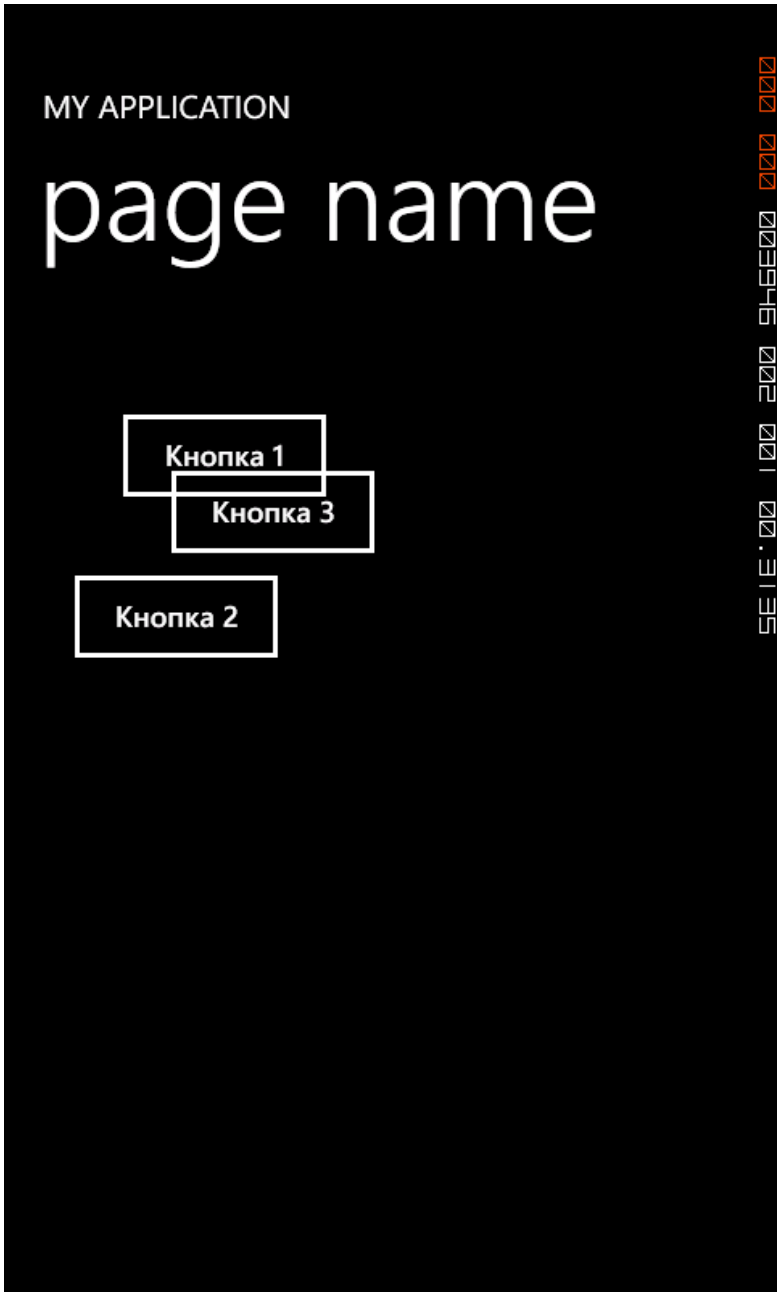
Элемент **Canvas** – предоставляет наиболее простой вариант разметки. Он может быть использован для абсолютного позиционирования элементов с использованием координат. Мы позиционируем элементы на Canvas, используя прикрепленные свойства (*AttachedProperties*). Прикрепленные свойства позволяют родительскому элементу расширять свойства размещенного на нем элемента управления (в нашем случае Button). Мы можем разместить несколько кнопок Button на Canvas, например, таким образом:

```
<Canvas>
  <Button Canvas.Top="50" Canvas.Left="50" Content="Кнопка 1" FontSize="18" Width="150" Height="75" />
  <Button Canvas.Top="150" Canvas.Left="20" Content="Кнопка 2" FontSize="18" Width="150" Height="75" />
  <Button Canvas.Top="85" Canvas.Left="80" Canvas.ZIndex="-1" Content="Кнопка 3" FontSize="18" Width="150" Height="75" />
</Canvas>
```

При запуске приложения (F5) увидим:



От новичка к эксперту



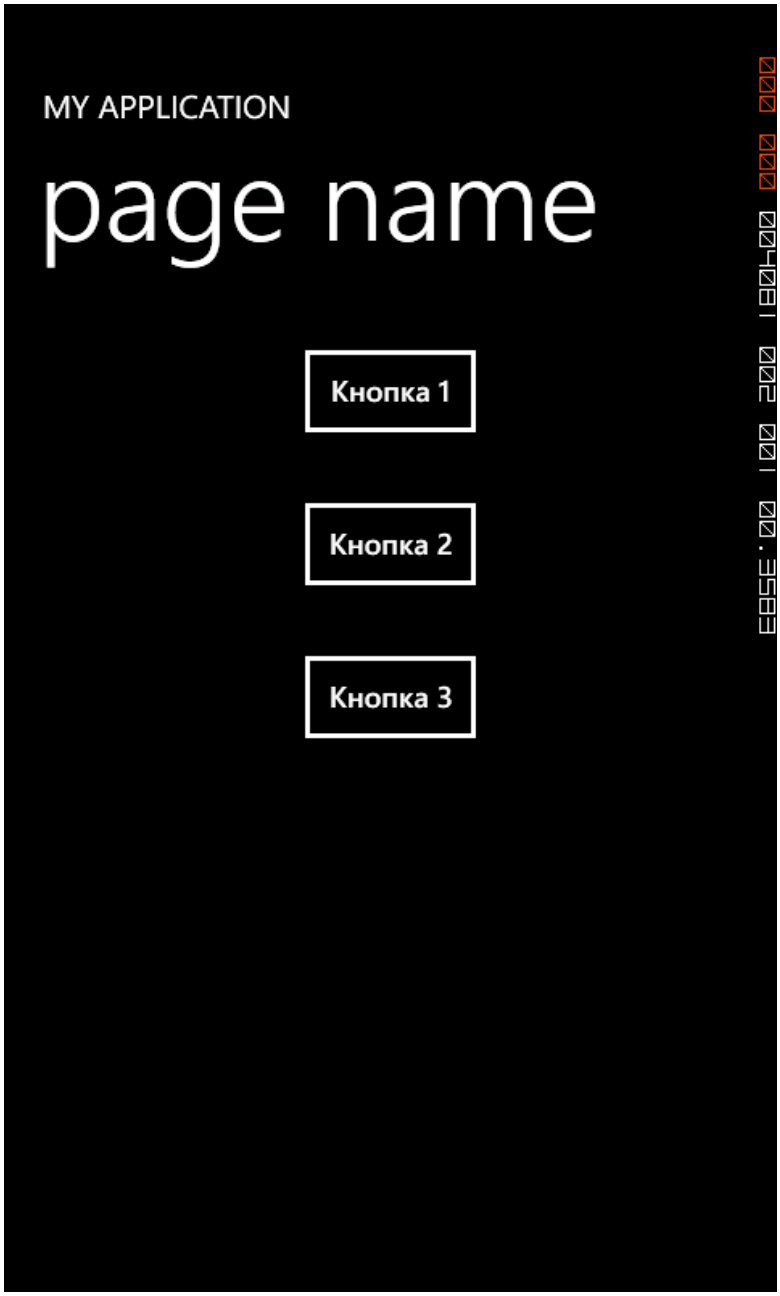
Как видно, элементы позиционированы абсолютным образом. Обратите внимание, что для одной из нопок указано прикрепленное свойство ZIndex, которое указывает как один элемент перекрывает другой элемент. Элемент Canvas полезен, когда элементы внутри не должны перемещаться, окно приложения менять ориентацию, а также, когда вам необходимо по тем или иным причинам позиционировать элементы интерфейса приложения абсолютным образом. В противном случае, работать с Canvas может быть сложнее, чем с такими элементами, как StackPanel или Grid.

StackPanel

Элемент **StackPanel** – предоставляет вариант разметки, элементы, помещенный в который, располагаются в стек горизонтально или вертикально (по умолчанию – вертикально). Сделаем пример из трех кнопок Button:

```
<StackPanel>
    <Button Margin="10" Content="Кнопка 1" FontSize="18" Width="130" Height="75" />
    <Button Margin="10" Content="Кнопка 2" FontSize="18" Width="130" Height="75" />
    <Button Margin="10" Content="Кнопка 3" FontSize="18" Width="130" Height="75" />
</StackPanel>
```

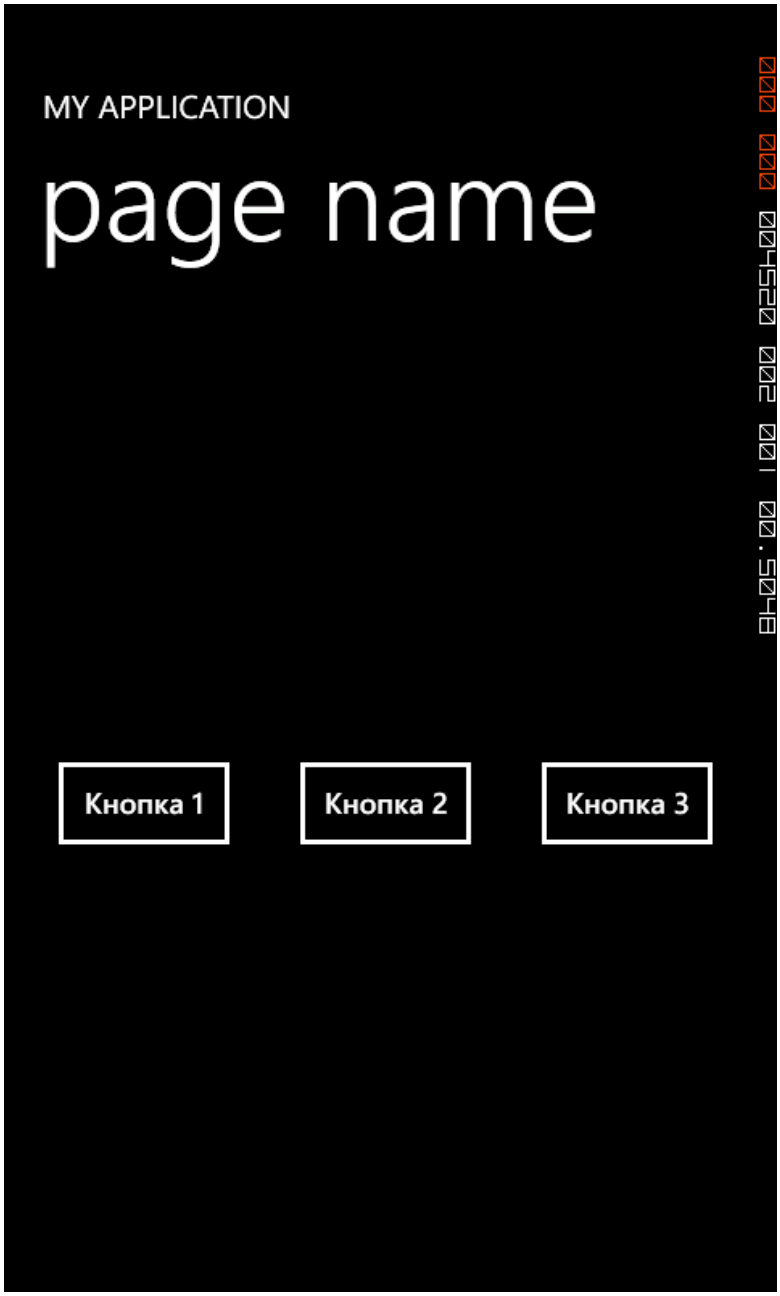
При запуске приложения (F5) увидим:



Или, если изменить ориентацию на горизонтальную (отличие в коде – только атрибут Orientation элемента StackPanel):

```
<StackPanel Orientation="Horizontal">
    <Button Margin="10" Content="Кнопка 1" FontSize="18" Width="130" Height="75" />
    <Button Margin="10" Content="Кнопка 2" FontSize="18" Width="130" Height="75" />
    <Button Margin="10" Content="Кнопка 3" FontSize="18" Width="130" Height="75" />
</StackPanel>
```

При запуске приложения (F5) увидим:



Элемент StackPanel предоставляет простой способ разместить элементы один за другим вертикально или горизонтально, без указания позиционирования элемента внутри контейнера.

Grid

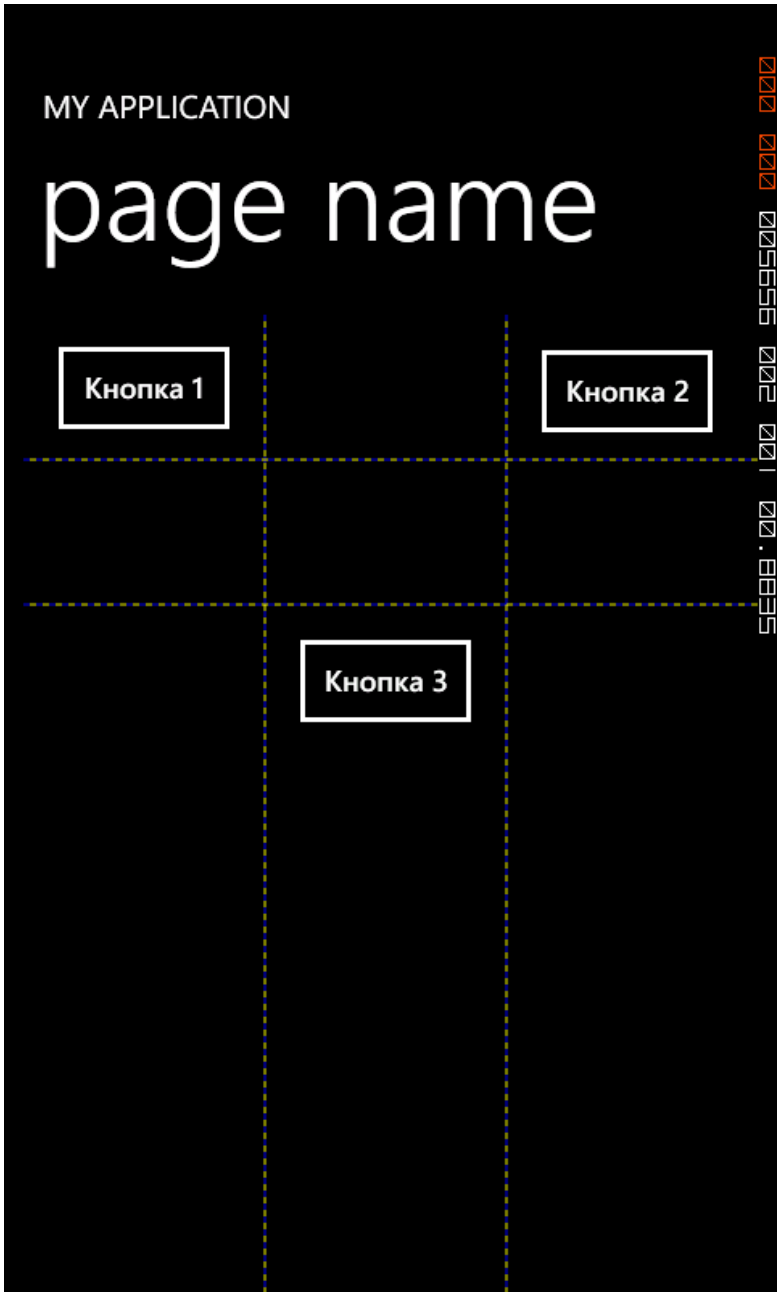
Элемент **Grid** – позволят позиционировать элементы внутри себя максимально гибко. Элемент Grid предоставляет возможность размещать элементы, используя строки и столбцы. Использование XAML элемента Grid отличается от использования элемента <table> при веб-разработке, где контент располагается внутри тегов <tr> и <td>. Разработчик определяет общую структуру сетки Grid, а потом используется присоединенные свойства, чтобы указать, где размещаются элементы.

Рассмотрим следующий код (обратите внимание, для облегчения понимания, включено отображение сетки, чего в рабочем коде, обычно, не предполагается):

```
<Grid ShowGridLines="True">
  <Grid.RowDefinitions>
       <RowDefinition Height="90"/>
       <RowDefinition Height="90"/>
       <RowDefinition Height="90"/>
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
       <ColumnDefinition Width="150" />
       <ColumnDefinition Width="150" />
       <ColumnDefinition Width="150" />
```

```
12.    </Grid.ColumnDefinitions>
13.
14.    <Button Grid.Column="0" Grid.Row="0" Content="Кнопка 1" FontSize="18" Width="130" Height="7
     5" />
15.    <Button Grid.Column="2" Grid.Row="0" Margin="10" Content="Кнопка 2" FontSize="18" Width="13
     0" Height="75" />
16.    <Button Grid.Column="1" Grid.Row="2" Margin="10" Content="Кнопка 3" FontSize="18" Width="13
     0" Height="75" />
17.
18. </Grid>
```

Мы определили элемент Grid с 3 строками и 3 столбцами с определенной шириной и высотой. Далее, элементы кнопки Button позиционируются внутри элемента Grid с использованием присоединённых свойств. Результат запуска программы будет выглядеть следующим образом:



Обратите внимание, как присоединенные свойства кнопки Button (Grid.Column и Grid.Row) указывают, где кнопка располагается в контейнере.

В работе с дизайном интерфейса большую помощь может оказать визуальный редактор, встроенный в Visual Studio, и Expression Blend. Оба редактора позволяют визуально отредактировать определения столбцов и строк, генерируя XAML-код.

Варианты разметки и элементы управления Pivot и Panorama

Платформа Windows Phone имеет доступные только для неё варианты разметки, представленные

следующими элементами управления:

- Pivot
- Panorama

Эти элементы управления представляют собой часть Metro-дизайна, о котором кратко упоминалось в первой части.
Это настолько важные варианты разметки, что в поставке средств разработки присутствует два специальных шаблона: Windows Phone Pivot Application и Windows Phone Panorama Application. Мы ещё к ним вернёмся, сейчас же попробуем добавить элементы управления Pivot и Panorama в уже существующее приложение ExploreBaseControls.

Pivot

Двойным щелчком по файлу MainPage.xaml перейдем к его редактированию. В редакторе XAML кода удалим всё содержимое основного элемента Grid с x:Name LayoutRoot:

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">
      <Grid.RowDefinitions>
           <RowDefinition Height="Auto"/>
           <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">



        </Grid>
    </Grid>
```

Так, что XAML код станет выглядеть следующим образом:

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">

</Grid>
```

Теперь нам надо добавить элемент управления Pivot. По умолчанию, он не включен в Toolbox с элементами управления в Visual Studio, также он не добавлен в доступное XAML пространство имен в используемом нами шаблоне.

Исправим это, одновременно, научившись добавлять новые пространства имен в XAML файл.

Чтобы добавлять элемент Pivot на страницу приложения, надо добавить в проект ссылку на библиотеку с этим элементом управления: Microsoft.Phone.Controls

Перейдите к заголовку XAML файла MainPage.xaml:

```
<phone:PhoneApplicationPage
    x:Class="ExploreBaseControls.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True" >
```

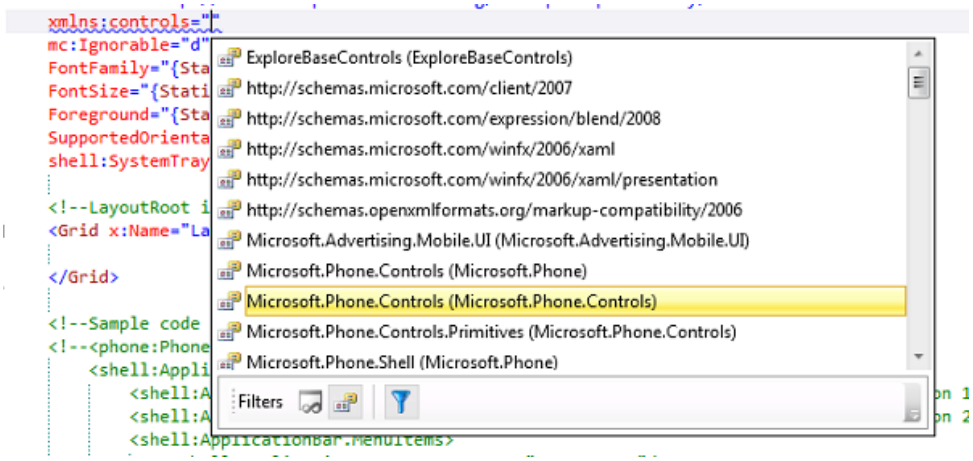
После строчки

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

нажмите Enter и в новой строке введите:

xmlns:controls=

система IntelliSense добавит кавычки и отобразит список доступных ссылок:



выберите Microsoft.Phone.Controls (Microsoft.Phone.Cotrols).

В результате строчка станет выглядеть следующим образом:

xmlns:controls="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"

и корневой XAML элемент файла MainPage.xaml станет выглядеть следующим образом:

```
<phone:PhoneApplicationPage
    x:Class="ExploreBaseControls.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:controls="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True" >
```

Сохраните файл и соберите проект (Ctrl+Shift+B). Это нужно, чтобы новое пространство имён гарантировано добавилось в систему IntelliSense.

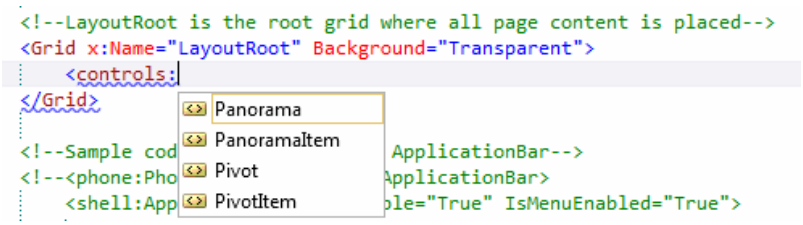
Теперь мы готовы добавлять элемент управления и разметки Pivot в наше приложение.

Двойным щелчком по файлу MainPage.xaml перейдем к его редактированию. В редакторе XAML кода перейдем к основному элементу Grid с x:Name LayoutRoot, который должен выглядеть следующим образом:

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">

</Grid>
```

Внутри этого элемента напишем:

<controls:
Система IntelliSense покажет нам, какие элементы доступны:

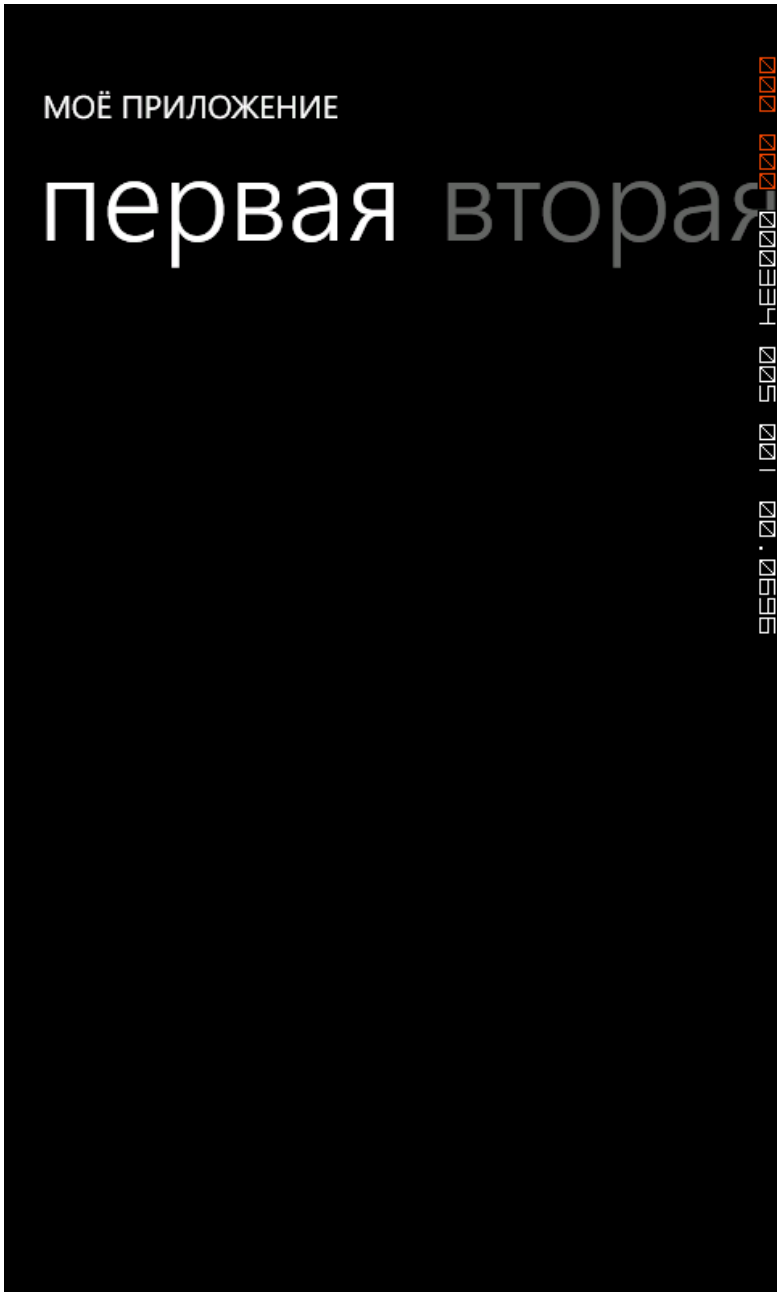


Мы видим, что в этом пространстве имён доступны элементы управления Pivot и Panorama и вспомогательные элементы PivotItem и PanoramaItem.

Добавим в XAML код элемент управления Pivot и внутрь его 3 элемента управления PivotItem, способом описаным выше, так, что XAML код будет выглядет следующим образом:

```
<controls:Pivot Title="МОЁ ПРИЛОЖЕНИЕ">
    <controls:PivotItem Header="первая">

    </controls:PivotItem>

    <controls:PivotItem Header="вторая">

    </controls:PivotItem>

    <controls:PivotItem Header="третья">

    </controls:PivotItem>
 </controls:Pivot>
```

Запустите приложение (F5) и посмотрите, как работает элемент управления Pivot.



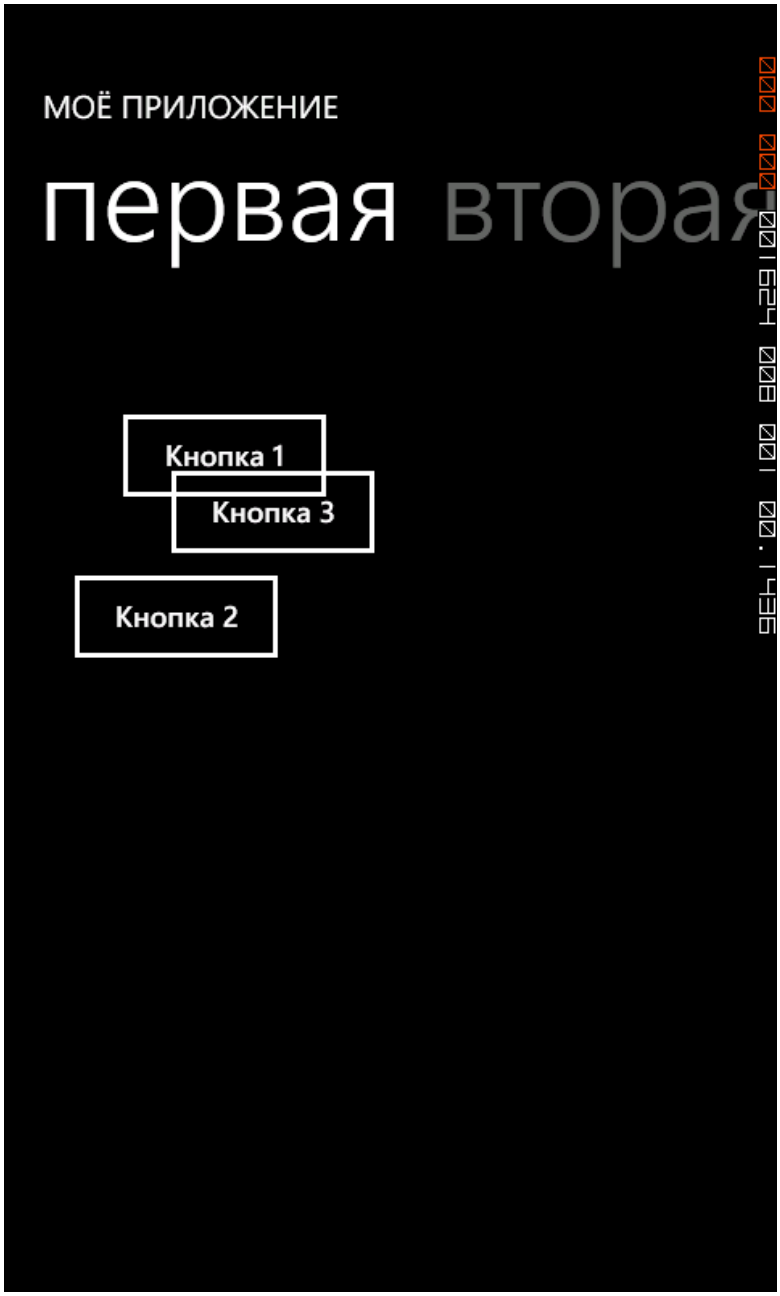
Каждый элемент управления PivotItem представляет из себя отдельную страницу, с соответствующим ограничением по размерам на содержание, на которую можно перейти прокруткой или щесчком/тапом по заголовку.

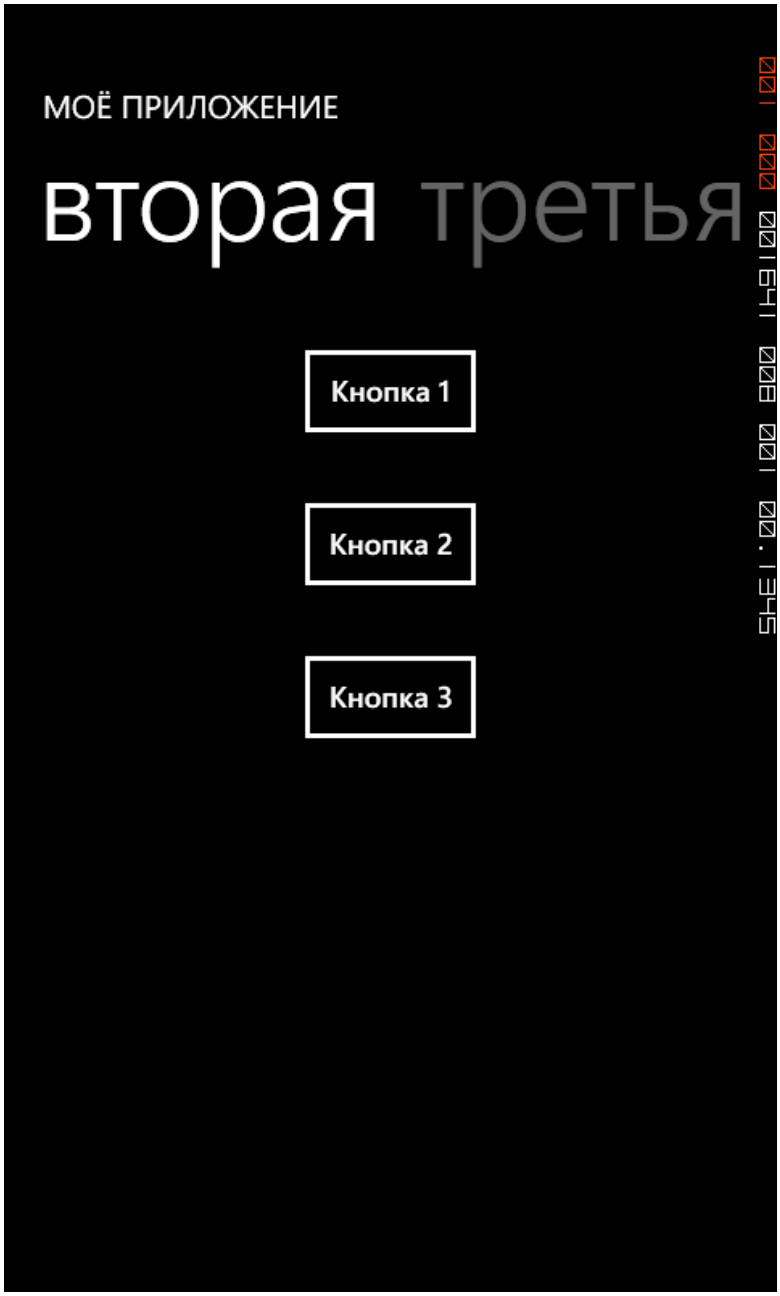
Добавим на каждую из трех страниц ранее уже использующийся нами XAML код с разметкой Canvas, StackPanel и Grid соответственно. Таким образом код XAML будет выглядеть следующим образом:

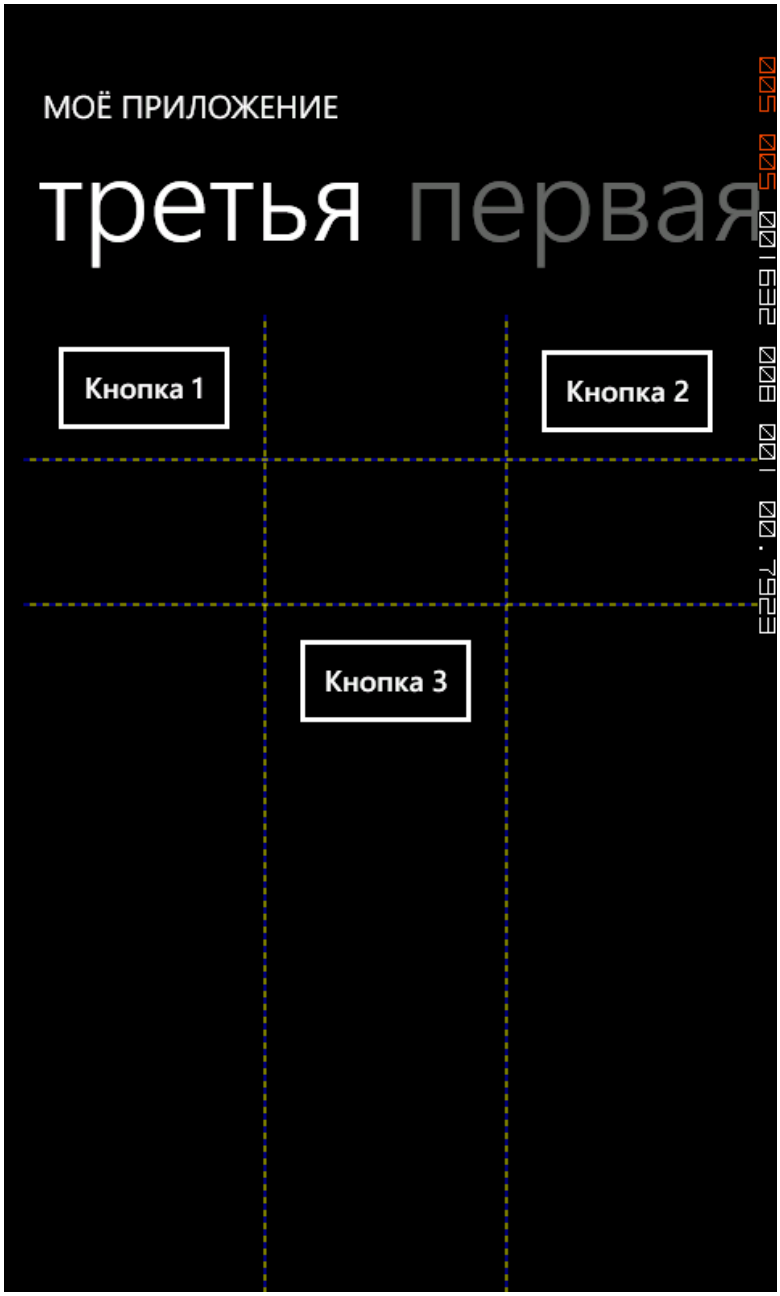
```
<controls:Pivot Title="МОЁ ПРИЛОЖЕНИЕ">
            <controls:PivotItem Header="первая">
                <Canvas>
                    <Button Canvas.Top="50" Canvas.Left="50" Content="Кнопка 1" FontSize="18" Width="150" Height="75" />
                    <Button Canvas.Top="150" Canvas.Left="20" Content="Кнопка 2" FontSize="18" Width="150" Height="75" />
                    <Button Canvas.Top="85" Canvas.Left="80" Canvas.ZIndex="-1" Content="Кнопка 3" FontSize="18" Width="150" Height="75" />
                </Canvas>
            </controls:PivotItem>

            <controls:PivotItem Header="вторая">
                <StackPanel>
                    <Button Margin="10" Content="Кнопка 1" FontSize="18" Width="130" Height="75" />
                    <Button Margin="10" Content="Кнопка 2" FontSize="18" Width="130" Height="75" />
```

```
14.                     <Button Margin="10" Content="Кнопка 3" FontSize="18" Width="130" Height="
    75" />
15.                 </StackPanel>
16.             </controls:PivotItem>
17.
18.             <controls:PivotItem Header="третья">
19.                 <Grid ShowGridLines="True">
20.                     <Grid.RowDefinitions>
21.                         <RowDefinition Height="90"/>
22.                         <RowDefinition Height="90"/>
23.                         <RowDefinition Height="90"/>
24.                     </Grid.RowDefinitions>
25.
26.                     <Grid.ColumnDefinitions>
27.                         <ColumnDefinition Width="150" />
28.                         <ColumnDefinition Width="150" />
29.                         <ColumnDefinition Width="150" />
30.                     </Grid.ColumnDefinitions>
31.
32.                     <Button Grid.Column="0" Grid.Row="0" Content="Кнопка 1" FontSize="18" Wid
    th="130" Height="75" />
33.                     <Button Grid.Column="2" Grid.Row="0" Margin="10" Content="Кнопка 2" FontS
    ize="18" Width="130" Height="75" />
34.                     <Button Grid.Column="1" Grid.Row="2" Margin="10" Content="Кнопка 3" FontS
    ize="18" Width="130" Height="75" />
35.
36.                 </Grid>
37.             </controls:PivotItem>
38.         </controls:Pivot>
```

Запустите приложение (F5) и посмотрите, как работает комбинация элемента управления/разметки Pivot с другими элементами управления/разметки.


МОЁ ПРИЛОЖЕНИЕ
первая
вторая
Кнопка 1
Кнопка 3
Кнопка 2


МОЁ ПРИЛОЖЕНИЕ
вторая
третья
Кнопка 1
Кнопка 2
Кнопка 3



Итак, мы научились добавлять элемент управления Pivot и протестировали комбинацию элемента управления/разметки Pivot с «классическими» элементами управления/разметки, одновременно, создав «классическое» Pivot-приложение, которое представляет одни и те же данные разными способами.

Panorama

Двойным щелчком по файлу MainPage.xaml перейдем к его редактированию. В редакторе XAML кода удалим всё содержимое основного элемента Grid с x:Name LayoutRoot, как в предыдущем случае, для элемента управления Pivot.

Поскольку мы уже добавили необходимое пространство имён в XAML файл, можно сразу добавить элемент управления Panorama и 3 элемента управления PanoramaItem внуть пустого теперь элемента Grid с x:Name LayoutRoot, который должен выглядеть перед добавлением кода, следующим образом:

```
1. <!--LayoutRoot is the root grid where all page content is placed-->
2. <Grid x:Name="LayoutRoot" Background="Transparent">
3.
4. </Grid>
```

После добавления элементов управления, то же место в XAML коде должно выглядеть следующим образом:

```
1. <!--LayoutRoot is the root grid where all page content is placed-->
```

```
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="моё приложение">
            <controls:PanoramaItem Header="первое">

            </controls:PanoramaItem>

            <controls:PanoramaItem Header="второе">

            </controls:PanoramaItem>

            <controls:PanoramaItem Header="третье">

            </controls:PanoramaItem>
        </controls:Panorama>
    </Grid>
```

Запустите приложение (F5) и посмотрите, как работает элемент управления Panorama.

Чтобы завершить общий визуальный стиль приложения на базе элемента управления/разметки Panorama давайте добавим общее фоновое изображение.

Для этого, добавьте в проект графический файл с расширением PNG размером 1024x768 с названием PanoramaBackground.png, а затем добавьте в файл MainPage.xaml сразу же после

<controls:Panorama Title="моё приложение">

следующий XAML код, устанавливающий фоновым изображением панорамы файл PanoramaBackground.png:

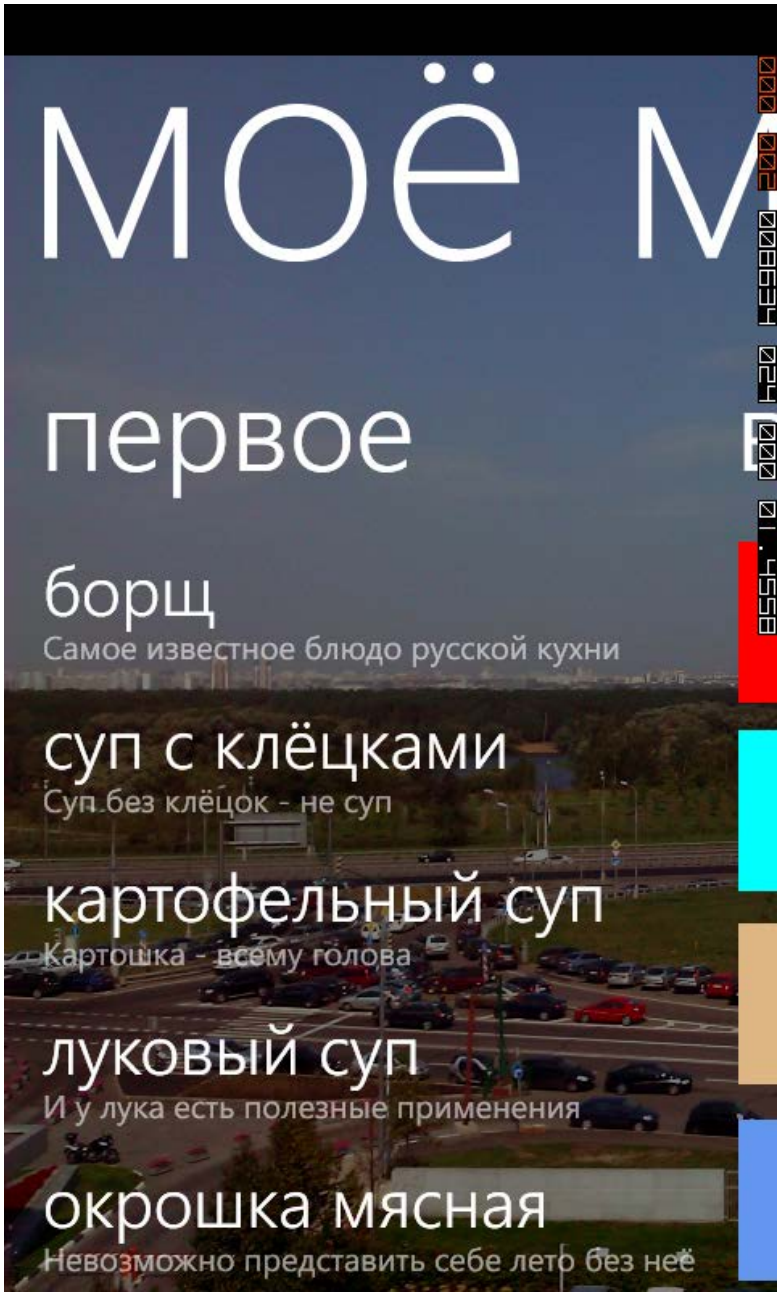
```
1. <controls:Panorama.Background>
2.       <ImageBrush ImageSource="PanoramaBackground.png"/>
3. </controls:Panorama.Background>
```

Запустите приложение (F5) и посмотрите, как работает элемент управления Panorama с установленным фоновым изображением.

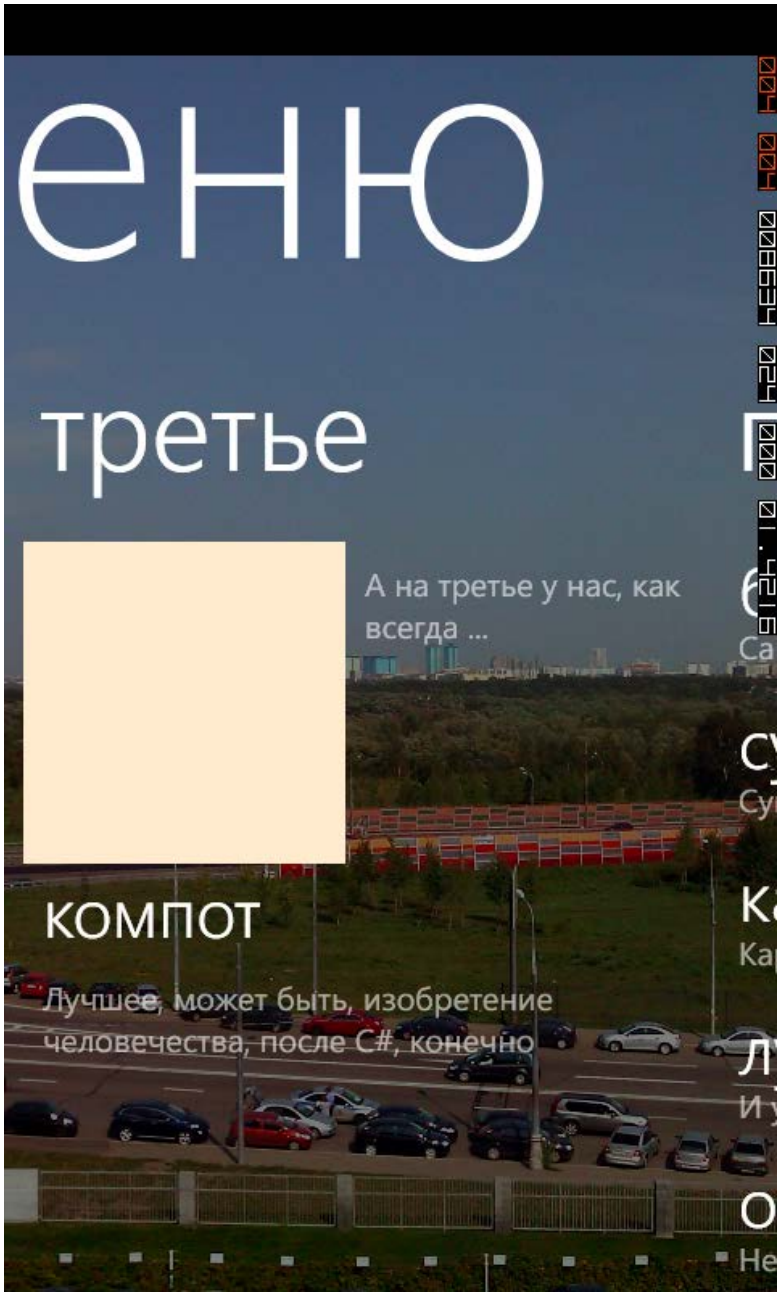
Чтобы сделать хороший пример приложения с использованием элемента управления/разметки Panorama правильным образом, удобно воспользоваться возможностями Expression Blend по созданию шаблонов представления и генерации примеров данных для режима дизайна, но эта тема выходит за рамки данного цикла статей. Курсы и пошаговые руководства по Expression доступны по ссылке: http://www.microsoft.com/design/toolbox/

Для простоты понимания мы просто добавим элементы управления в PanoramaItem, чтобы эмулировать пример приложения панорамы. Дизайн элементов возьмём из шаблона приложения панорамы и заполним текстом, исходя из концепции приложения-меню русского ресторана.

Можете сделать это самостоятельно или посмотреть на код, который идет вместе с этой статьёй. Если вы добавите дизайн творчески, то у вас при запуске приложения, получится что-то вроде:


моё м
первое
борщ
Самое известное блюдо русской кухни
суп с клёцками
Суп без клёцок - не суп
картофельный суп
Картошка - всему голова
луковый суп
И у лука есть полезные применения
окрошка мясная
Невозможно представить себе лето без неё


ё мен
второе
жаль, что нету фотографии
долма
Младшие братики голубцов
шашлык
Предположительно из баранины
ризотто
Надеюсь, я правильно вспомнил название
помидоры



Основные элементы управления

Итак, мы рассмотрели основные элементы управления, которые являются элементами разметки. Рассмотрим остальные основные элементы управления, доступные разработчику на Silverlight под Windows Phone. С некоторыми из них мы уже знакомы, так как активно использовали в наших примерах приложений.

Элемент управления	Краткое описание
Border	Предоставляет другому элементу управления рамку и/или фоновое изображение.
Button	Кнопка, при нажатии пользователем генерирует событие Click.
CheckBox	Представляет собой элемент управления – флажок. Может быть установлен или снят, опционально можно включить поддержку «неопределенного» состояния.
HyperlinkButton	Кнопка, отображающая ссылку. При нажатии переходи на ссылку, указанную в свойстве NavigateUri.
Image	Позволяет отобразить картинку.
ListBox	Отображает список элементов, которые могут выбираться пользователем. Контент может задаваться разнообразным количеством способов.
MediaElement	Позволяет проиграть аудио или видео.

PasswordBox	Специальный элемент для ввода пароля, скрывает ввод пользователя.
ProgressBar	Отображает текущий прогресс пользователю.
RadioButton	Позволяет пользователю выбрать только один вариант из нескольких. Сгруппированные элементы управления (один GroupName) позволяют выбрать только один вариант из группы.
ScrollViewer	Добавляет возможность прокручивания дочерним элементам.
Slider	Позволяет пользователю выбрать из нескольких последовательных вариантов. Позиция соотносится со значением свойства Value.
TextBlock	Позволяет отобразить простой текст, без возможности редактирования.
TextBox	Используется для ввода как короткого, так и многострочного текста.
Map	Отображает карту Bing
WebBrowser	Отображает отрендеренный HTML

Теперь посмотрим, как эти элементы выглядят, какие у них есть свойства. В нашем приложении ExploreBaseControls на странице MainPage.xaml удалите всё, что мы вставляли ранее внутрь элемента Grid:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
```

После этого, воспользовавшись Toolbox перетяните элементы управления с него на дизайн-представление телефона, исследуйте их свойства в окне свойств, попробуйте собрать приложение и посмотреть, как они работают.

Для примера ниже приведён XAML код

```
<!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <ScrollViewer>
            <StackPanel>
                <Border Margin="10" Width="100" Height="100" BorderThickness="2" CornerRadius="5" BorderBrush="{StaticResource PhoneAccentBrush}"/>
                <ProgressBar Name="MyProgressBar" Maximum="10" ValueChanged="MyProgressBar_ValueChanged"/>
                <Slider Name="MySlider" ValueChanged="MySlider_ValueChanged"/>
                <Button Margin="10" Width="150" Height="100" Content="Button" Name="MyButton" Click="MyButton_Click" />
                <CheckBox Margin="10" Content="CheckBox" Height="70" HorizontalAlignment="Center" Name="MyCheckBox" IsThreeState="True"  />
                <HyperlinkButton Margin="10" Content="HyperlinkButton" Height="30" Name="MyHyperlinkButton" />
                <Image Margin="10" Height="150" Name="MyImage" Stretch="Fill" Width="200" Source="/ExploreBaseControls;component/PanoramaBackground.png" />
                <ListBox Name="MyListBox">
                    <TextBlock Margin="10 0 0 0">Строка 1</TextBlock>
                    <TextBlock Margin="20 0 0 0">Строка 2</TextBlock>
                    <TextBlock Margin="40 0 0 0">Строка 3</TextBlock>
                    <TextBlock Margin="10 0 0 0">Строка 4</TextBlock>
                    <TextBox Height="70" Width="300" Name="MyTextBox" Margin="0 10 0 0"/>
                </ListBox>
                <PasswordBox Name="MyPasswordBox"/>
                <RadioButton GroupName="MyGroup" Content="Опция 1"/>
                <RadioButton GroupName="MyGroup" Content="Опция 2"/>
                <RadioButton GroupName="MyGroup" Content="Опция 3"/>
                <RadioButton GroupName="MyGroup" Content="Опция 4"/>
            </StackPanel>
        </ScrollViewer>
    </Grid>
```

И код, добавленный в файл MainPage.xaml.cs

```
private void MyButton_Click(object sender, RoutedEventArgs e)
{
     MyProgressBar.Value += 1;
}

private void MySlider_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
```

```
 8.          MyProgressBar.Value = MySlider.Value;
 9. }
10.
11. private void MyProgressBar_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
     e)
12. {
13.       MySlider.Value = MyProgressBar.Value;
14. }
```

Если установить Silverlight for Windows Phone Toolkit и добавить его в проект, то разработчик сможет использовать дополнительно следующие элементы управления:

Элемент управления	Краткое описание
AutoCompleteBox	Представляет собой TextBox с возможностью автоматического дополнения строки ввода по указанному списку.
ListPicker	Комбинация TextBox, отображающего текущее значение с выпадающим списком из которого можно значение выбрать.
LongListSelector	Расширенная версия ListBox с групировкой и виртуализацией
ContextMenu	Контекстное меню.
DatePicker	Выбор даты.
TimePicker	Выбор времени.
ToggleSwitch	Текст с переключателем вкл/выкл
WrapPanel	Элемент разметки, реализующий перенос для дочерних элементов.

Текстовые поля и контекст ввода

Ввод текста – важная и, может быть, не самая приятная часть взаимодействия пользователя с телефоном. Чтобы несколько упростить этот процесс, а также уменьшить количество ошибок ввода уже давно предлагается множество разнообразных технологий. Один из вариантов – указание контекста ввода для текстовых полей, чтобы платформа отображала соответствующий элемент управления.

Создайте новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и назовите его ExploreTextContexts. Обратите внимание, что весь код, для простоты, будем вставлять в страницу MainPage.xaml внутрь элемента Grid:

```
1. <!--ContentPanel - place additional content here-->
2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
```

Всего поддерживается более 60 контекстов ввода, полный список можно посмотреть по следующей ссылке: http://msdn.microsoft.com/en-us/library/system.windows.input.inputscopenamevalue(VS.95).aspx

Ниже табличка с наиболее часто используемыми контекстами ввода с описанием:

Контекст ввода	Описание
по умолчанию	Стандартная клавиатура. Первая буква предложение автоматически устанавливается в заглавную.
Text	Предоставляет автодополение, автокоррекцию, автоматическую расстановку апострофов, автоматическую установку акцентов и заглавных букв.
Number	Цифры
TelephoneNumber	Отображает телефонную клавиатуру
EmailSmtpAddress	Добавляет символ @ для быстрого доступа
CurrencyAmountAndSymbol	Цифры и значки валют
URL	Отключается вся автокоррекция, добавляются кнопки .com и Go

Добавьте в наше приложение, внутрь элемента Grid следующий XAML код:

```
1. <ScrollViewer>
2.             <StackPanel>
3.             <TextBlock>По умолчанию</TextBlock>
4.             <TextBox/>
```

```
5.
6.              <TextBlock>Текст</TextBlock>
7.              <TextBox InputScope="Text"/>
8.
9.              <TextBlock>Цифры</TextBlock>
10.             <TextBox InputScope="Number"/>
11.
12.             <TextBlock>Номер телефона</TextBlock>
13.             <TextBox InputScope="TelephoneNumber"/>
14.
15.             <TextBlock>E-mail</TextBlock>
16.             <TextBox InputScope=" EmailSmtpAddress "/>
17.
18.             <TextBlock>Поиск</TextBlock>
19.             <TextBox InputScope="Search"/>
20.
21.             <TextBlock>Деньги</TextBlock>
22.             <TextBox InputScope="CurrencyAmountAndSymbol"/>
23.
24.             </StackPanel>
25.             </ScrollViewer>
```

Запустите приложение (F5) и проверьте, как меняется панель ввода и настройки автокоррекции и автодополнения для разных контекстов ввода. Переключите эмулятор на русский интерфейс, добавьте русскую клавиатуру и протестируйте приложение ещё раз.

Итоги и следующие шаги

Итак, мы познакомились с вариантами элементов разметки, доступными в Windows Phone 7, уделяя особое внимание Pivot и Panorama, познакомились с основными элементами управления и узнали, что такое контекст ввода.

На следующем шаге мы познакомимся с дополнительными возможностями платформы, которые может использовать разработчик мобильных приложений.

Файлы для загрузки

Проект ExploreBaseControls с основными элементами управления
Проект ExploreBaseControls с элементом управления Pivot
Проект ExploreBaseControls с элементом управления Panorama
Проект ExploreTextContext

Разработка под Windows Phone: Часть 3: Использование возможностей платформы

Рейтинг

Мобильная платформа – это не только элементы разметки и управления. Платформа Windows Phone позволяет разработчику максимально воспользоваться своими программными возможностями через задачи выбора и запуска, элементы управления Map и WebBrowser, также предоставляя доступ к таким своим аппаратным возможностям, как акселерометр и гелокационным данным.

Задачи запуска и выбора

Для начала определимся с теоретической частью. Для доступа к программным возможностям платформы у разработчика есть задачи запуска и выбора. Работа с ними в несколько отличается со стороны разработчика.

Задачи запуска (launchers)

Для того, чтобы использовать любую из задач запуска разработчику необходимо выполнить следующие шаги:

1. Создать экземпляр класса задачи запуска.
2. Установить свойства полученного объекта.
3. Вызвать метод **Show** объекта.

Разработчику доступны следующие задачи запуска:

Задача запуска	Краткое описание
BingMapsDirectionsTask	Запускается приложение Bing Maps и отображается путь между двумя точками. Может указываться одна точка или обе. В случае указания одной точки за вторую будет взято текущее местоположение пользователя.
BingMapsTask	Запускается приложение Bing Maps, можно передать строчку поиска.
ConnectionSettingsTask	Отображается интерфейс сетевых настроек.
EmailComposeTask	Запускается e-mail приложение с отображением интерфейса создания нового сообщения. Позволяет пользователю посылать e-mail из ваших приложений.
MarketplaceDetailTask	Запускается клиентское приложение Windows Phone Marketplace с отображением страницы детальных сведений определенного приложения.
MarketplaceHubTask	Запускается клиентское приложение Windows Phone Marketplace.
MarketplaceReviewTask	Запускается клиентское приложение Windows Phone Marketplace с отображением страницы обзора определенного приложения
MarketplaceSearchTask	Запускается клиентское приложение Windows Phone Marketplace и отображаются результаты поиска по указанному поисковому запросу.
MediaPlayerLauncher	Запускается медиаплеер.
PhoneCallTask	Запускается приложение Phone. Позволяет пользователю звонить из вашего приложения.
SearchTask	Запускается приложение поиска.
ShareLinkTask	Позволяет пользователю опубликовать ссылку в указанной социальной сети.
ShareStatusTask	Позволяет пользователю опубликовать сообщение в указанной социальной сети.
SmsComposeTask	Запускается приложение Messaging с отображением интерфейса создания нового сообщения.
WebBrowserTask	Запускается веб-браузер.

Создайте новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и

От новичка к эксперту

назовите его ExploreLaunchers. Обратите внимание, что весь код, для простоты, будем вставлять в страницу MainPage.xaml внутрь элемента Grid:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
```

Добавьте код в XAML файл, чтобы он выглядел следующим образом:

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel Orientation="Horizontal">
                <StackPanel Width="240">
                    <Button Name="EMailMe" Width="190" Height="75" Content="E-mail" Click="EMailMe_Click"/>
                    <Button Name="AppDetails" Width="190" Height="75" Content="Программа" Click="AppDetails_Click"/>
                    <Button Name="MarketHub" Width="190" Height="75" Content="Рынок" Click="MarketHub_Click" />
                    <Button Name="MarketSearch" Width="190" Height="75" Content="Поиск по ..." Click="MarketSearch_Click"/>
                    <Button Name="MediaPlayer" Width="190" Height="75" Content="Плеер" Click="MediaPlayer_Click"/>
                </StackPanel>
                <StackPanel Width="240">
                    <Button Name="PhoneCall" Width="190" Height="75" Content="Звонок" Click="PhoneCall_Click"/>
                    <Button Name="Search" Width="190" Height="75" Content="Искать" Click="Search_Click"/>
                    <Button Name="SendSms" Width="190" Height="75" Content="SMS" Click="SendSms_Click"/>
                    <Button Name="WebBrowse" Width="190" Height="75" Content="Веб" Click="WebBrowse_Click"/>
                </StackPanel>
            </StackPanel>
        </Grid>
    </Grid>
```

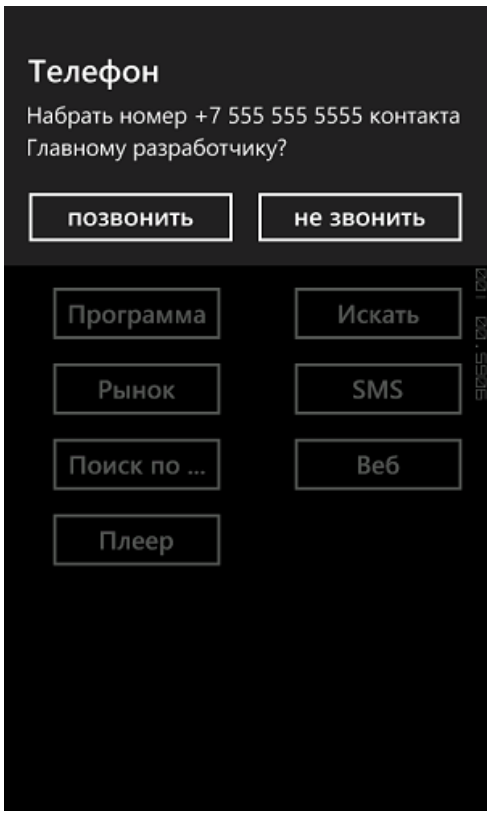
Добавьте в блок using следующий модуль.

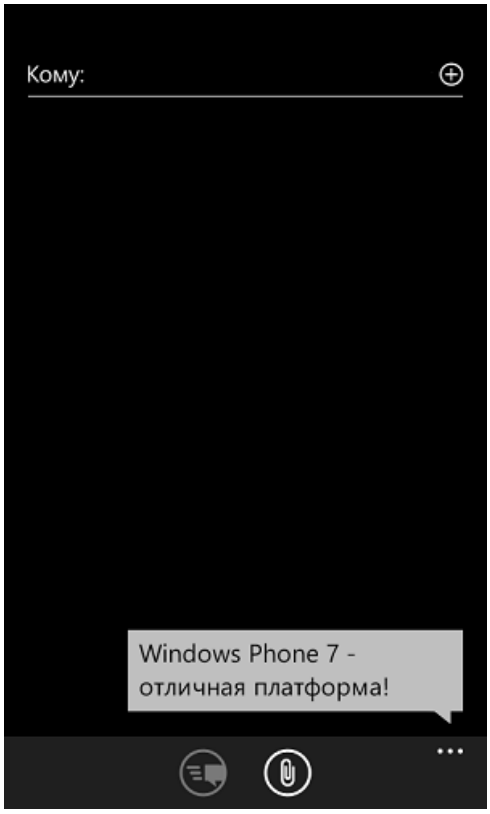
```
using Microsoft.Phone.Tasks;

Сразу после конструктора MainPage добавьте следующий код:
private void EMailMe_Click(object sender, RoutedEventArgs e)
        {
            EmailComposeTask compose = new EmailComposeTask();
            compose.To = "rush4apps@microsoft.com";
            compose.Body = "Крутая инициатива! Я хочу участовать! Продлите акцию, пожлайста!";
            compose.Show();
        }

        private void AppDetails_Click(object sender, RoutedEventArgs e)
        {
            //MarketplaceDetailTask marketDetails = new MarketplaceDetailTask();
            //marketDetails.ContentType = MarketplaceContentType.Applications;
            //marketDetails.ContentIdentifier = "введите сюда идентфикатор приложения";
            //marketDetails.Show();
        }

        private void MarketHub_Click(object sender, RoutedEventArgs e)
        {
            MarketplaceHubTask marketHub = new MarketplaceHubTask();
            marketHub.ContentType = MarketplaceContentType.Applications;
            marketHub.Show();
        }

        private void MarketSearch_Click(object sender, RoutedEventArgs e)
        {
            MarketplaceSearchTask marketSearch = new MarketplaceSearchTask();
            marketSearch.ContentType = MarketplaceContentType.Applications;
            marketSearch.SearchTerms = "GPSInfo";
            marketSearch.Show();
        }

        private void MediaPlayer_Click(object sender, RoutedEventArgs e)
```

```
36.         {
37.             //MediaPlayerLauncher player = new MediaPlayerLauncher();
38.             //player.Controls = MediaPlaybackControls.Stop;
39.             //player.Media = "укажите URI с которого будет играть контент";
40.             //player.Show();
41.         }
42.
43.         private void PhoneCall_Click(object sender, RoutedEventArgs e)
44.         {
45.             PhoneCallTask call = new PhoneCallTask();
46.             call.DisplayName = "Главному разработчику";
47.             call.PhoneNumber = "+7 555 555 5555";
48.             call.Show();
49.         }
50.
51.         private void Search_Click(object sender, RoutedEventArgs e)
52.         {
53.             SearchTask search = new SearchTask();
54.             search.SearchQuery = "rush4apps";
55.             search.Show();
56.         }
57.
58.         private void SendSms_Click(object sender, RoutedEventArgs e)
59.         {
60.             SmsComposeTask sms = new SmsComposeTask();
61.             sms.Body = "Windows Phone 7 - отличная платформа!";
62.             sms.Show();
63.         }
64.
65.         private void WebBrowse_Click(object sender, RoutedEventArgs e)
66.         {
67.             WebBrowserTask web = new WebBrowserTask();
68.             web.Uri = new Uri("http://msdn.com/ru-ru/");
69.             web.Show();
70.         }
```

Мы добавили кнопок по количеству демонстрируемых задач загрузки и в обработчике соответствующей кнопки вызываем интерфейс задачи загрузки. Некоторые из задач загрузки будут работать только на реальном устройстве. Обработчик с закомментированным кодом требуют вашего вмешательства перед тем, как их можно будет раскомментировать и запустить.

Соберите и запустите приложение (F5), посмотрите, как работают задачи запуска.





В качестве самостоятельного упражнения попробуйте добавить кнопки и дописать код для оставшихся задач запуска.

Задачи выбора (choosers)

Для того, чтобы использовать любую из задач выбора разработчику необходимо выполнить следующие шаги:

1. Создать экземпляр класса задачи запуска.
2. Добавить обработчик события **Completed**.

3. Установить свойства полученного объекта.
4. Вызвать метод **Show** объекта.
5. Получить данные от задачи выбора в обработчике события **Completed**.

Разработчику доступны следующие задачи выбора:

Задача выбора	Краткое описание
AddressChooserTask	Запускает приложение Contacts. Используется, чтобы получить адрес контакта, выбранного пользователем.
CameraCaptureTask	Запускает приложение Camera. Используется, чтобы дать возможность пользователю сделать фотографию из вашего приложения.
EmailAddressChooserTask	Запускает приложение Contacts. Используется, чтобы получить e-mail адрес контакта, выбранного пользователем.
GameInviteTask	Отображает экран приглашения в многопользовательскую сессию игры.
PhoneNumberChooserTask	Запускает приложение Contacts. Используется, чтобы получить телефон контакта, выбранного пользователем.
PhotoChooserTask	Запускает приложение PhotoChooser. Используется, чтобы дать возможность пользователю выбрать картинку.
SaveContactTask	
SaveEmailAddressTask	Запускает приложение Contacts. Используется, чтобы сохранить e-mail адрес в новый или существующий контакт.
SavePhoneNumberTask	Запускает приложение Contacts. Используется, чтобы сохранить номер телефона в новый или существующий контакт.
SaveRingtoneTask	Позволяет пользователю сохранить аудиофайл как системный рингтон.

Создайте новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и назовите его ExploreChoosers. Обратите внимание, что весь код, для простоты, будем вставлять в страницу MainPage.xaml внутрь элемента Grid:

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
```

Добавьте код в XAML файл, чтобы он выглядел следующим образом:

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <Button Width="190" Height="75" Name="Camera" Content="Камера" Click="Camera_Click"/>
                <Button Width="190" Height="75" Name="Email" Content="EMail" Click="Email_Click"/>
                <Button Width="190" Height="75" Name="Phone" Content="Телефон" Click="Phone_Click"/>
                <Button Width="190" Height="75" Name="ChoosePhoto" Content="Фото" Click="ChoosePhoto_Click"/>
                <Button Width="190" Height="75" Name="SaveEMail" Content="Сохр. EMail" Click="SaveEMail_Click"/>
                <Button Width="190" Height="75" Name="SavePhone" Content="Сохр. тел." Click="SavePhone_Click"/>
            </StackPanel>
        </Grid>
```

Добавьте в блок using следующий модуль.

```
using Microsoft.Phone.Tasks;
```

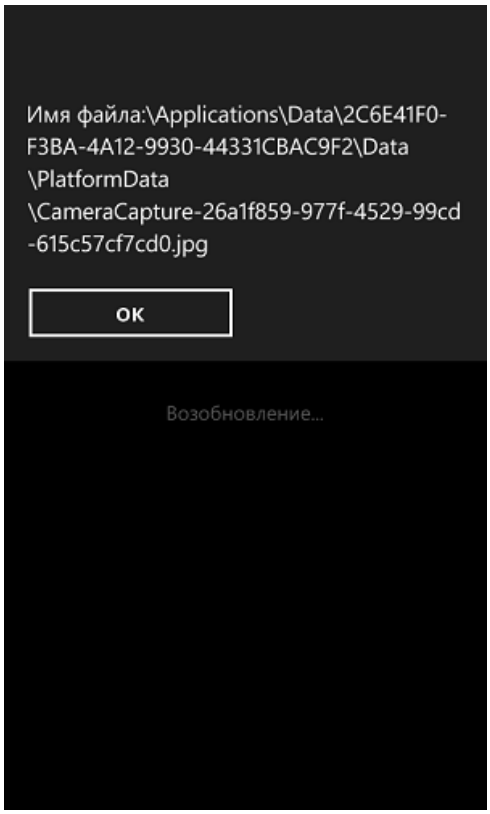
Сразу после конструктора MainPage добавьте следующий код:

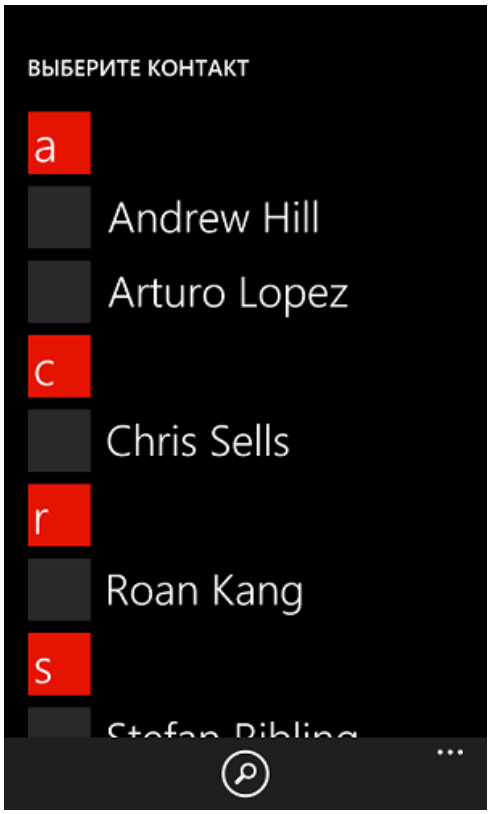
```
private void Camera_Click(object sender, RoutedEventArgs e)
        {
            CameraCaptureTask camera = new CameraCaptureTask();
```

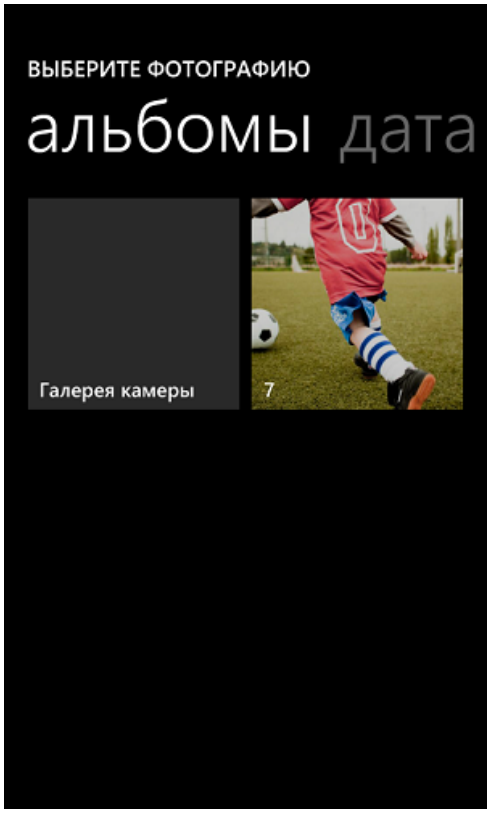
```
            camera.Completed += new EventHandler<PhotoResult>(camera_Completed);
            camera.Show();
        }

        void camera_Completed(object sender, PhotoResult e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show("Имя файла:" + e.OriginalFileName);
        }

        private void Email_Click(object sender, RoutedEventArgs e)
        {
            EmailAddressChooserTask email = new EmailAddressChooserTask();
            email.Completed += new EventHandler<EmailResult>(email_Completed);
            email.Show();
        }

        void email_Completed(object sender, EmailResult e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show(e.Email);
        }

        private void Phone_Click(object sender, RoutedEventArgs e)
        {
            PhoneNumberChooserTask phone = new PhoneNumberChooserTask();
            phone.Completed += new EventHandler<PhoneNumberResult>(phone_Completed);
            phone.Show();
        }

        void phone_Completed(object sender, PhoneNumberResult e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show(e.PhoneNumber);
        }

        private void ChoosePhoto_Click(object sender, RoutedEventArgs e)
        {
            PhotoChooserTask photo = new PhotoChooserTask();
            photo.Completed += new EventHandler<PhotoResult>(photo_Completed);
            photo.Show();
        }

        void photo_Completed(object sender, PhotoResult e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show(e.OriginalFileName);
        }

        private void SaveEMail_Click(object sender, RoutedEventArgs e)
        {
            SaveEmailAddressTask saveEmail = new SaveEmailAddressTask();
            saveEmail.Completed += new EventHandler<TaskEventArgs>(saveEmail_Completed);
            saveEmail.Email = "rush4apps@microsoft.com";
            saveEmail.Show();
        }

        void saveEmail_Completed(object sender, TaskEventArgs e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show("EMail сохранен!");
        }

        private void SavePhone_Click(object sender, RoutedEventArgs e)
        {
            SavePhoneNumberTask savePhone = new SavePhoneNumberTask();
            savePhone.Completed += new EventHandler<TaskEventArgs>(savePhone_Completed);
            savePhone.PhoneNumber = "+7 495 555 5555";
            savePhone.Show();
        }

        void savePhone_Completed(object sender, TaskEventArgs e)
        {
            if (e.TaskResult == TaskResult.OK)
                MessageBox.Show("Номер сохранен!");
        }
```

Мы добавили кнопок по количеству демонсрируемых задач выбора и в обработчике соответствующей кнопки регистрируем обработчик события Completed и вызываем интерфейс задачи выбора.

Соберите и запустите приложение (F5), посмотрите, как работают задачи выбора.





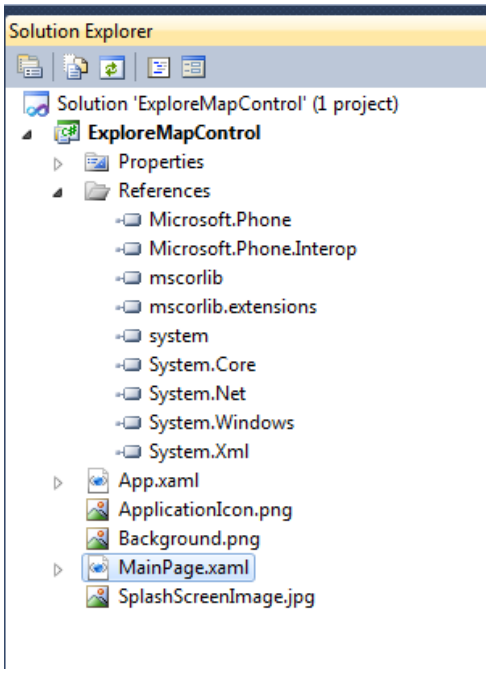


В качестве самостоятельного упражнения попробуйте добавить кнопки и дописать код для оставшихся задач выбора.

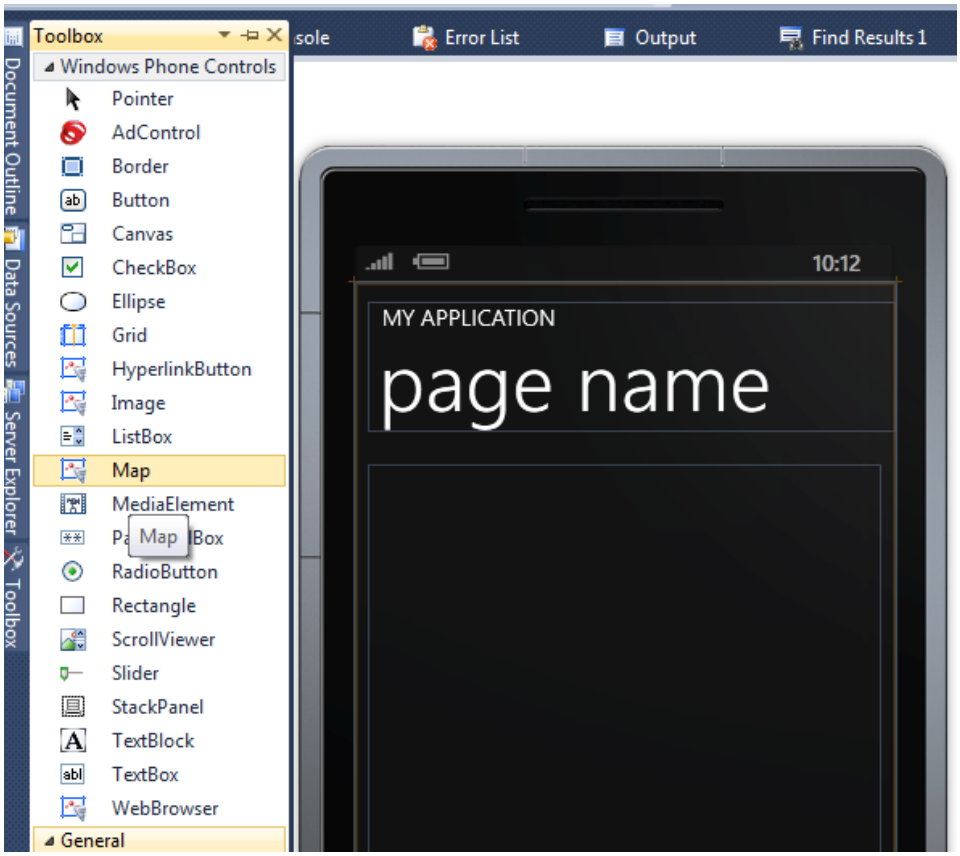
Элемент управления Map

Чтобы познакомиться с элементом управления Map, давайте создадим новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и назовём его ExploreMapControl.

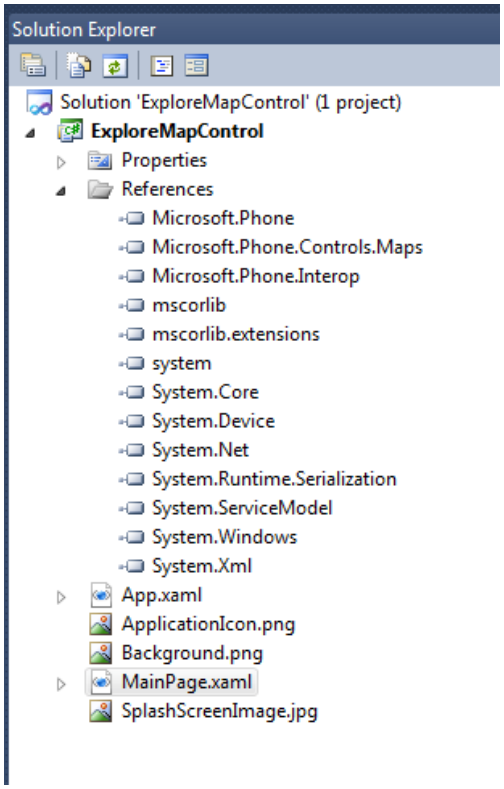
После того, как создадите проект, посмотрите, на какие библиотеки он ссылается:



Разверните Toolbox, если он свёрнут, и перетяните элемент управления Map в дизайнер интерфейса приложения



Обратите внимание, что теперь проект ссылается на Microsoft.Phone.Controls.Map:



Двойным щелчком перейдем к странице MainPage.xaml и посмотрим, что изменилось в XAML коде.

Добавился элемент управления Map из пространства имён my:

```
1. <my:Map Height="50" HorizontalAlignment="Left" Margin="201,198,0,0" Name="map1" VerticalAlig
   nment="Top" Width="100" />
```

Посмотрев на заголовок XAML документа можно увидеть, что это на пространство имён:

```
xmlns:my="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
```

Давайте заменим my на map, чтобы название пространства имён соответствовало его содержанию:

```
xmlns:map="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
```

и

```
<map:Map Height="50" HorizontalAlignment="Left" Margin="201,198,0,0" Name="map1" VerticalAlignment="Top" Width="100" />
```

Отредактируем XAML код элемента управления Map или воспользуемся панелью Properites так, чтобы элемент занимал большую часть свободного пространства и переименуем элемент в MyMap:

<map:Map Name="MyMap"/>

Запустите приложение (F5) и посмотрите, как выглядит элемент управления во время исполнения.

Обратили внимание на белый баннер в центре экрана, который говорит, что у нас неправильные авторизационные данные? Это потому что этот элемент управления использует сервис карт от Bing и для его использования требуется регистрация. Зарегистрироваться и получить ключ можно на портале Bing Maps: http://www.bingmapsportal.com/

В завершение регистрации разработчик получает строковый ключ, который надо указать в свойстве CredentialsProvider элмента управления, также его можно вынести в ресурсы или данные.

Добавим простые элементы управления картой: уменьшение/увеличение масштаба, смена режима отображения карты:

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <map:Map Name="MyMap">
                <Button Name="ZoomIn" Content="+" HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="60" Width="60" Margin="-100,0,0,-5" Click="ZoomIn_Click" FontWeight="Bold" Padding="0,-9,0,0"/>

                <Button Name="ZoomOut" Content="-" HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="60" Width="60" Margin="100,0,0,-5" Click="ZoomOut_Click" FontWeight="Bold" Padding="0,-9,0,0" />
```

```
 7.
 8.                <Button Name="LayoutChange" Content="L" HorizontalAlignment="Center" Vertica
    lAlignment="Bottom" Height="60" Width="60" Margin="0,0,0,-5" FontWeight="Bold" Padding="0,-
    9,0,0" Click="LayoutChange_Click"/>
 9.
10.             </map:Map>
11.         </Grid>
```

И обработаем эти события в коде приложения:

```
 1. private void ZoomIn_Click(object sender, RoutedEventArgs e)
 2.         {
 3.             MyMap.ZoomLevel += 1;
 4.         }
 5.
 6.         private void ZoomOut_Click(object sender, RoutedEventArgs e)
 7.         {
 8.             MyMap.ZoomLevel -= 1;
 9.         }
10.
11.         private void LayoutChange_Click(object sender, RoutedEventArgs e)
12.         {
13.             if (MyMap.Mode is RoadMode)
14.             {
15.                 MyMap.Mode = new AerialMode(true);
16.             }
17.             else
18.             {
19.                 MyMap.Mode = new RoadMode();
20.             }
21.
22.         }
```

Не забудьте добавить в блок using следующую директиву:

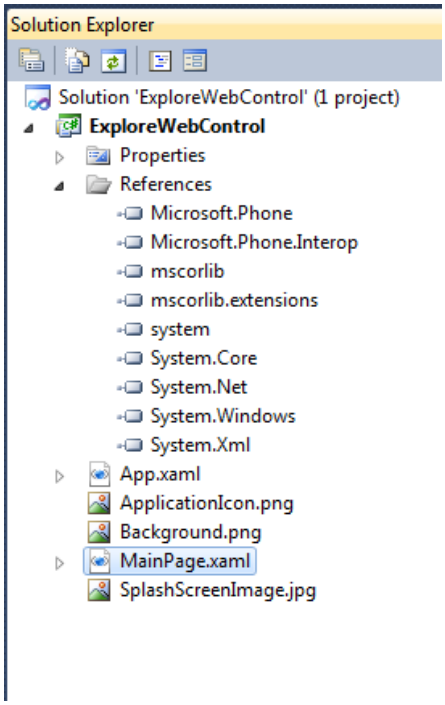
using Microsoft.Phone.Controls.Maps;
Запустите приложение (F5) и проверьте, что наши элементы управления работают, как предполагается.

В соответствии с Metro-дизайном панель с кнопками внизу окна приложения мы должны были бы оформить в виде Application Bar, только в целях упрощения примера мы используем более простой вариант. В качестве самостоятельного упражнения можете попробовать убрать кнопки, раскоментировать пример кода Application Bar в XAML файле и переделать приложение в соответствии со стилем Metro.

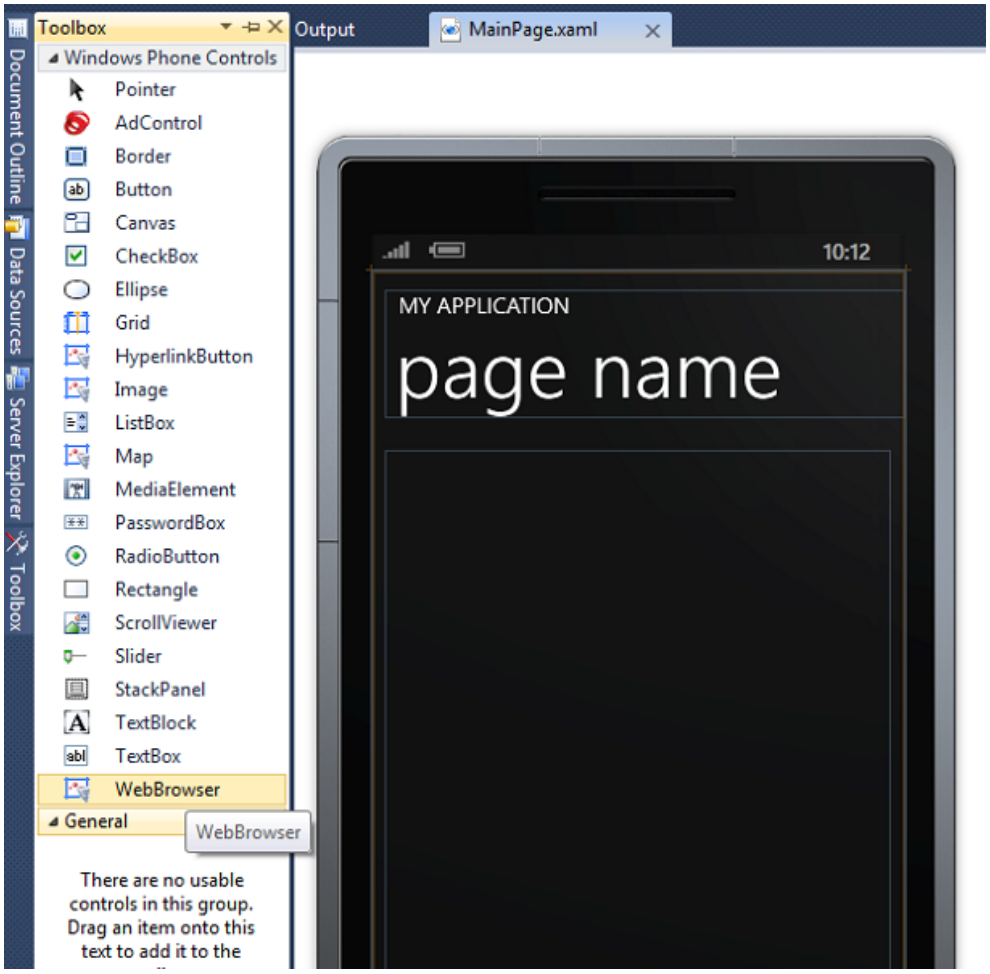
Элемент управления WebBrowser

Чтобы познакомиться с элементом управления Map, давайте создадим новый проект из шаблона Windows Phone Application, как мы это уже делали в первой части и назовём его ExploreWebControl.

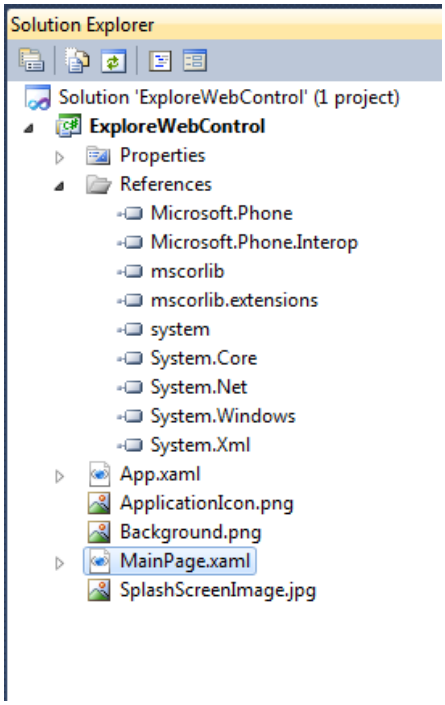
После того, как создадите проект, посмотрите, на какие библиотеки он ссылается:



Разверните Toolbox, если он свёрнут, и перетяните элемент управления WebBrowser в дизайнер интерфейса приложения



Обратите внимание, что никаких дополнительных ссылок не добавилось:



Двойным щелчком перейдем к странице MainPage.xaml и посмотрим, что изменилось в XAML коде.

```
<phone:WebBrowser HorizontalAlignment="Left" Margin="174,169,0,0" Name="webBrowser1" Vertica
lAlignment="Top" />
```

Используется пространство имён phone, которое было добавлено в шаблон по умолчанию:

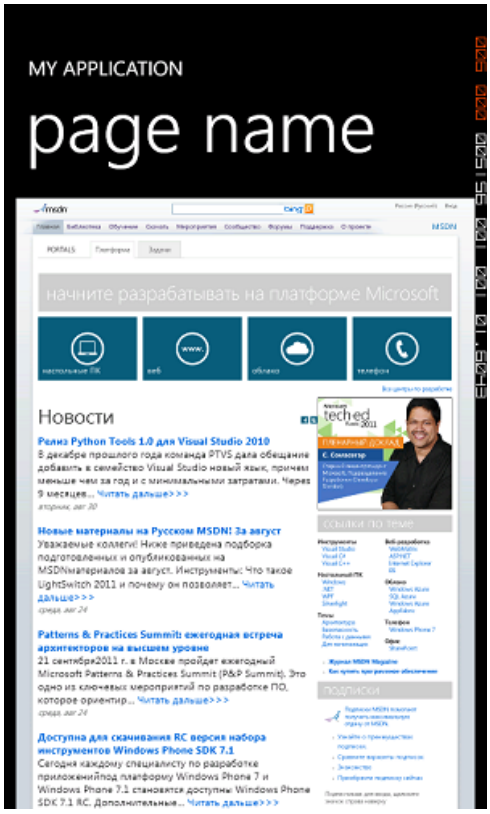
```
xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
```

Оно содержит корневой элемент нашего XAML файла – элемент управления страницу приложения:

```
<phone:PhoneApplicationPage
    x:Class="ExploreWebControl.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">
```

Отредактируем XAML код элемента управления WebBrowser или воспользуемся панелью Properites так, чтобы элемент занимал большую часть свободного пространства и переименуем элемент в MyBrowser, а также добавим прямо в XAML файл определение свойства Source, которое указывает элементу управления, откуда брать HTML для рендеренга. В качестве примера страницы я используют заглавную страницу русского MSDN:

<phone:WebBrowser Name="MyBrowser" Source="http://msdn.com/ru-ru/"/>
Запустите приложение (F5) и посмотрите, как выглядит элемент управления во время исполнения.



Сделаем теперь свой простой браузер: добавим строку ввода и кнопку Go, а также небольшой код для передачи URL браузеру и запуску навигации.

XAML код:

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <StackPanel Orientation="Horizontal">
                    <TextBox Name="Url" InputScope="URL" Width="380"/>
                    <Button Name="Go" Width="82" Content="Go" Click="Go_Click"/>
                </StackPanel>

                 <phone:WebBrowser Name="MyBrowser" Source="http://msdn.com/ru-
ru/" Height="530"/>
            </StackPanel>
        </Grid>
```

Обработчик события нажатия кнопки:

```
private void Go_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                Uri url = new Uri(Url.Text);
                MyBrowser.Navigate(url);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Запустите приложение (F5) и проверьте, что оно работает и даже выводит сообщение об ошибке, при вводе неправильного URL.

Акселерометр

От программных возможностей платформы доступных разработчику давайте перейдем к аппаратным.

Давайте откроем ранее созданное приложение ExploreMapControl и добавим в него возможности акселерометра.

Для работы с акселерометром к проекту надо подключить библиотеку Microsoft.Devices.Sensors. Для этого в Solution Explorer правым щелчком мыши по папке Refernce вызовите контекстное меню, выберите Add Reference и добавьте Microsoft.Devices.Sensors. Далее, в блок using добавим следующую директиву:

using Microsoft.Devices.Sensors;

Теперь мы готовы работать с акселерометром.

Сделаем простое приложение, которое при покачивании телефона выполняло масштабирование карты. При отклонении от себя масштаб будет увеличиваться, а при отклонении телефона на себя – уменьшаться.

Добавим глобальную переменную в класс представляющую акселерометр:

private Accelerometer myAccel;

В конструктор класса добавим инициализацию акселерометра, обработчик изменения его состояний и стартуем акселерометр:

myAccel = new Accelerometer();

myAccel.CurrentValueChanged += new EventHandler<SensorReadingEventArgs<AccelerometerReading>>(myAccel_CurrentValueChanged);

myAccel.Start();

Новый обработчик событий использует класс Vector3 из библиотеки Microsoft.Xna.Framework. Добавим в проект ссылку на библиотеку Microsoft.Xna.Framework, также как для Microsoft.Devices.Sensors, а также в блок using директиву:

using Microsoft.Xna.Framework;

Для того, чтобы реализовать тот функционал, который нам необходим, нужно хранить текущее (предыдущее состояние) акселерометра в классе. Добавим переменную в класс:

```
private Vector3 currentValues;
Теперь можно перейти к описанию логики обработчика состояния акселерометра:
void myAccel_CurrentValueChanged(object sender, SensorReadingEventArgs<AccelerometerReading>
e)
{
   if (myAccel.IsDataValid)
   {
      float deltaZ = (currentValues - e.SensorReading.Acceleration).Z;
      float Z = e.SensorReading.Acceleration.Z;

      currentValues = e.SensorReading.Acceleration;

      if (Z < 0 && deltaZ > 0)
      {
           //увеличиваем масштаб
      }
      if (Z > 0 && deltaZ < 0)
      {
           //уменьшаем масштаб
      }
    }

 }
```

Мы собираемся изменять состояние пользовательского интерфейса из потока акселерометра. Напрямую это невозможно. Функция обновления должна вызываться спциальным образом через объект Dispatcher:

Dispatcher.BeginInvoke
Создадим два обработчика в классе:

```
private void HandleZoomIn()
  {
      MyMap.ZoomLevel += 1;
  }

  private void HandleZoomOut()
  {
      MyMap.ZoomLevel -= 1;
  }
```

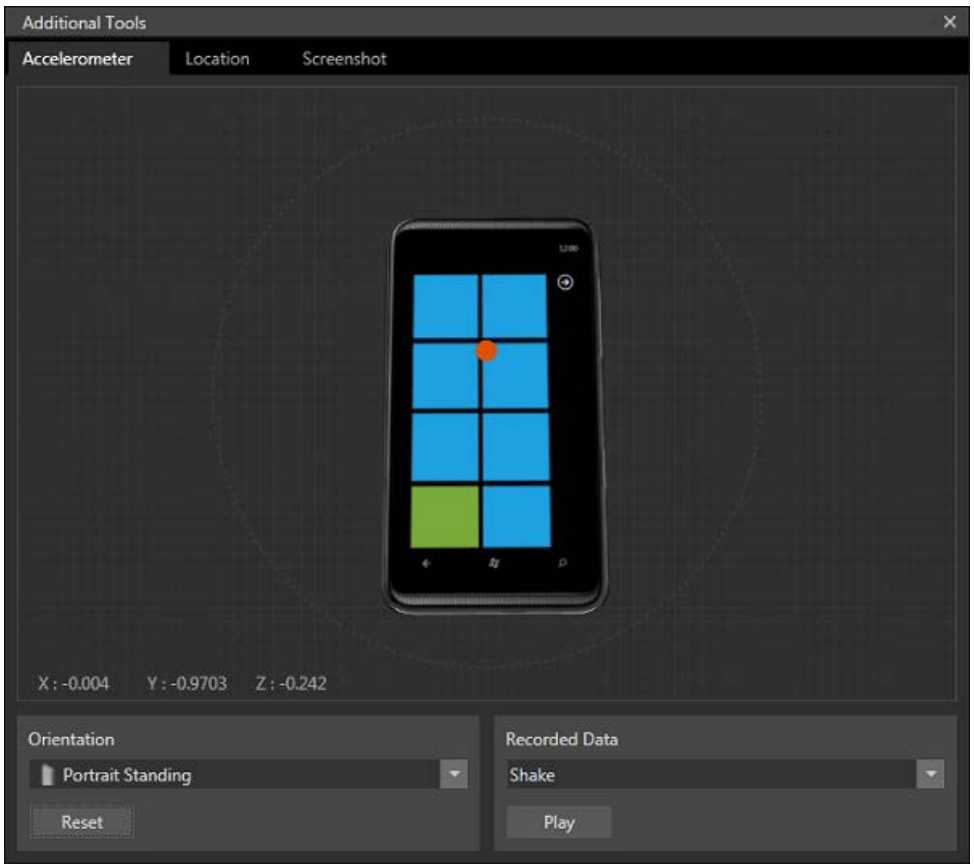
И добавим соответствующие вызовы через Dispatcher:

Dispatcher.BeginInvoke(() => HandleZoomIn());
и

Dispatcher.BeginInvoke(() => HandleZoomOut());
Результирующий класс будет выглядеть следующим образом:

```
 1. private Accelerometer myAccel;
 2.         private Vector3 currentValues;
 3. 
 4.         // Constructor
 5.         public MainPage()
 6.         {
 7.             InitializeComponent();
 8. 
 9.             myAccel = new Accelerometer();
10.             myAccel.CurrentValueChanged += new EventHandler<SensorReadingEventArgs<Acceleromet
    erReading>>(myAccel_CurrentValueChanged);
11. 
12.             myAccel.Start();
13.             currentValues = myAccel.CurrentValue.Acceleration;
14.         }
15. 
16. 
17. 
18.         void myAccel_CurrentValueChanged(object sender, SensorReadingEventArgs<AccelerometerR
    eading> e)
19.         {
20.             if (myAccel.IsDataValid)
21.             {
22.                 float deltaZ = (currentValues - e.SensorReading.Acceleration).Z;
23.                 float Z = e.SensorReading.Acceleration.Z;
24. 
25.                 currentValues = e.SensorReading.Acceleration;
26. 
27.                 if (Z < 0 && deltaZ > 0)
28.                 {
29.                     Dispatcher.BeginInvoke(() => HandleZoomIn());
30.                 }
31.                 if (Z > 0 && deltaZ < 0)
32.                 {
33.                     Dispatcher.BeginInvoke(() => HandleZoomOut());
34.                 }
35.             }
36. 
37.         }
38. 
39.         private void HandleZoomIn()
40.         {
41.             MyMap.ZoomLevel += 1;
42.         }
43. 
44.         private void HandleZoomOut()
45.         {
46.             MyMap.ZoomLevel -= 1;
47.         }
48. 
49.         private void ZoomIn_Click(object sender, RoutedEventArgs e)
50.         {
51.               MyMap.ZoomLevel += 1;
52.         }
53. 
54.         private void ZoomOut_Click(object sender, RoutedEventArgs e)
55.         {
56.             MyMap.ZoomLevel -= 1;
57.         }
58. 
59.         private void LayoutChange_Click(object sender, RoutedEventArgs e)
60.         {
61.             if (MyMap.Mode is RoadMode)
62.             {
63.                 MyMap.Mode = new AerialMode(true);
64.             }
65.             else
66.             {
67.                 MyMap.Mode = new RoadMode();
68.             }
69. 
70.         }
```

Запустите приложение (F5) и проверьте, что оно работает, как ожидается. Воспользуйтесь возможностями эмулятора, который поддерживает эмуляцию акселерометра.



Геолокационные данные

Наконец перейдём к геолокационным сервисам, доступным на телефоне. Сервис предоставляет информацию, используя комбинацию информации получаемой от Wi-Fi, сотовой связи и данных от GPS приёмника.

Давайте откроем ранее созданное приложение ExploreMapControl и добавим в него возможности предоставляемые сервисами геолокации.

Для начала, добавим в блок using следующую директиву:

using Microsoft.Devices.Sensors;
Теперь мы готовы работать с сервисами локаций/местоположения.

Сначала напишем простое дополнение к нашей программе, которое будет центрировать карту в соответствии с гелолокационными данными, полученными от сервисов.

Добавим в класс определение переменной типа GeoCoordinateWatcher, которая позволит нам инициализировать сервисы геолокации и полчать от них данные.

private GeoCoordinateWatcher myGeoWatcher;

В конструктор класса, сразу же после кода относящегося к акселерометру добавим код инициализации и регистрации на события изменения статуса сервисов (они могут быть недоступны, могут быть не готовы и т.д.) и события изменения положения.

```
1. yGeoWatcher = new GeoCoordinateWatcher();
2.             myGeoWatcher.MovementThreshold = 100.0f;
3.
4.             myGeoWatcher.StatusChanged += new EventHandler<GeoPositionStatusChangedEventArgs>(
   myGeoWatcher_StatusChanged);
5.
6.             myGeoWatcher.PositionChanged += new EventHandler<GeoPositionChangedEventArgs<GeoCo
   ordinate>>(myGeoWatcher_PositionChanged);
```

Хорошее приложение должно правильно обрабатывать статусы геосервисов, т.к. они не всегда могут выдавать данные и могут тратить достаточно большое время на инициализацию. Для начала мы просто оставим обработчик пустым, так как тестировать приложением мы будем на эмуляторе, и там эти проблемы отсутствуют.

Также правильнее будет поместить запуск сервиса геолокации в отдельный поток, чтобы не тормозить загрузку приложения, в нашем первом варианте приложения, с учётом использования эмулятора мы пока будем запускать сервис прямо в конструкторе класса:

myGeoWatcher.TryStart(false, TimeSpan.FromSeconds(60));

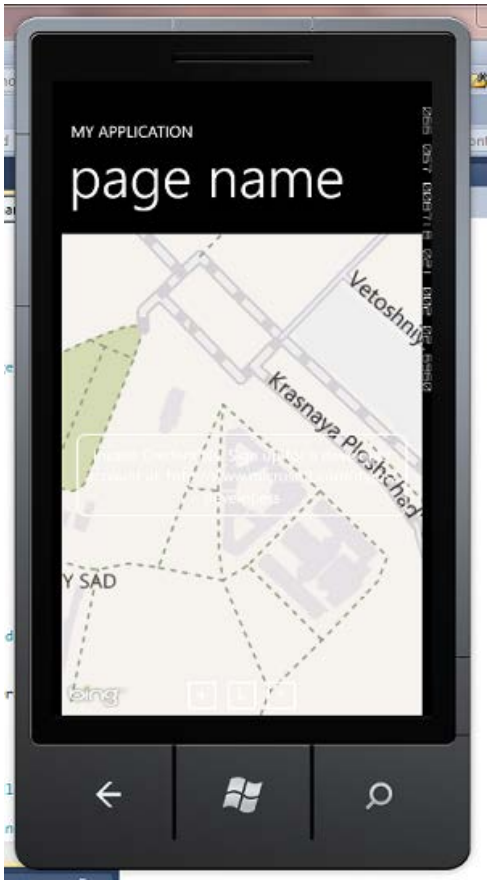
Если Visual Studio автоматически сгенерировала нам обработчики событий StatusChanged и PositionChanged, закомментируйте или сотрите код в этих методах, вызывающий исключение NotImplemented:

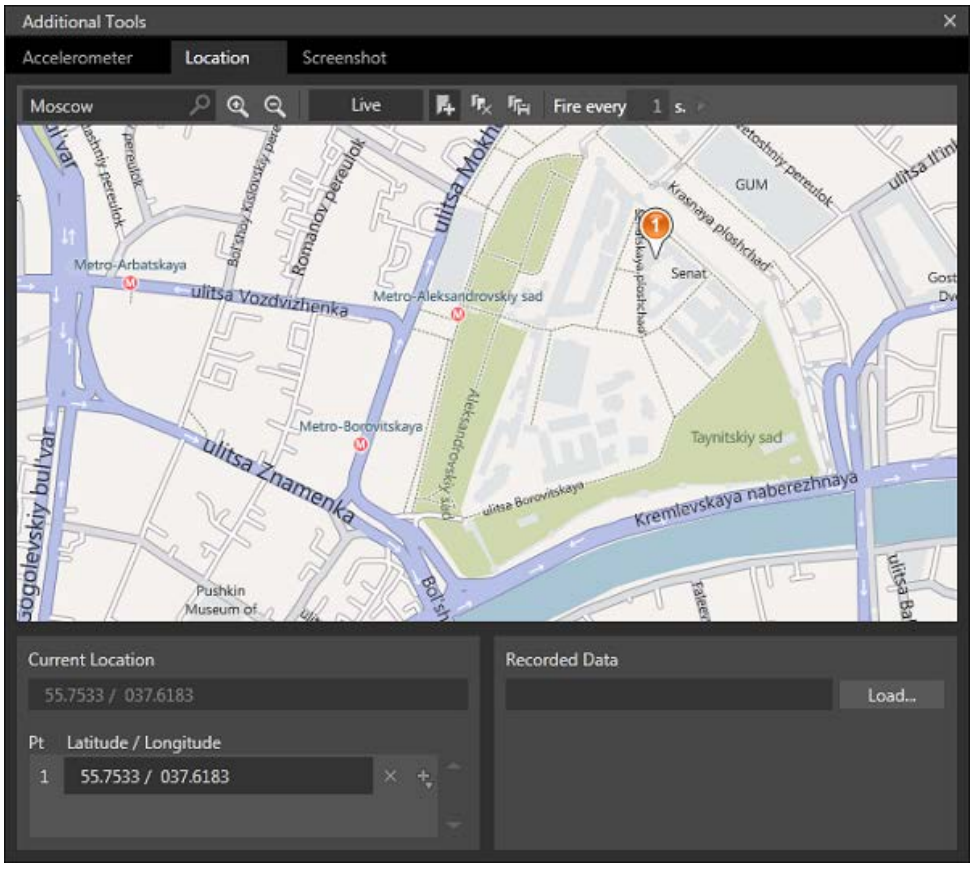
throw new NotImplementedException();

В обработчик события PositionChangedдобавьте код, центрирующий карту при изменении позиции:

```
1. void myGeoWatcher_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e
   )
2.         {
3.             MyMap.Center = e.Position.Location;
4.         }
```

Запустите приложение (F5) и воспользуйтесь возможностями эмулятора по эмуляции геолкационных данных, чтобы проверить работу программы. Увеличьте масштаб так, чтобы убедиться, что позиционирование происходи правильно.





Следующим шагом, улучшения нашей программы может стать запуск сервисов в другом потоке, добавление строки статуса геолокационных данных, а также создание на карте точки, отмечающей наше местоположение.

Для использования потоков, добавим в блок using следующую директиву:

using System.Threading;

В конструкторе до запуска сервисов добавим код:

new Thread(startMyGeoWotcher).Start();

После этого создадим функцию, не принимающую и не возвращающую значений, с именем startMyGeoWotcher и перенесём в неё код запуска сервисов:

```
void startMyGeoWotcher()
{
     myGeoWatcher.TryStart(false, TimeSpan.FromSeconds(60));
}
```

Теперь добавим элемент управление TextBlock для отображения статуса сервисов геолокации

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
            <TextBlock Name="GeoStatus" HorizontalAlignment="Center" VerticalAlignment="Top" Text="Geo Status .."  />
            <map:Map Name="MyMap" Height="580">

                <Button Name="ZoomIn" Content="+" HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="60" Width="60" Margin="-100,0,0,-5" Click="ZoomIn_Click" FontWeight="Bold" Padding="0,-9,0,0"/>

                <Button Name="ZoomOut" Content="-" HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="60" Width="60" Margin="100,0,0,-5" Click="ZoomOut_Click" FontWeight="Bold" Padding="0,-9,0,0" />

                <Button Name="LayoutChange" Content="L" HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="60" Width="60" Margin="0,0,0,-5" FontWeight="Bold" Padding="0,-9,0,0" Click="LayoutChange_Click"/>

            </map:Map>
            </StackPanel>
        </Grid>
```

И допишем вывод статусов в обработчик StatusChanged:

```
void myGeoWatcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case GeoPositionStatus.Disabled:
                    if (myGeoWatcher.Permission == GeoPositionPermission.Denied)
                    {
                        GeoStatus.Text = "Сервис выключен";
                    }
                    else
                    {
                        GeoStatus.Text = "На этом устройстве сервис недоступен";
                    }
                    break;
                case GeoPositionStatus.Initializing:
                    GeoStatus.Text = "Сервис инициализируется";
                    break;
                case GeoPositionStatus.NoData:
                    GeoStatus.Text = "Данные о месположении недоступны";
                    break;
                case GeoPositionStatus.Ready:
                    GeoStatus.Text = "Данные о местоположении доступны";
                    break;
            }

        }
```

Наконец, в обработчик события изменения позиции добавим установку точки на карте.

В класс добавим переменную типа Pushpin:

private Pushpin myPushpin;

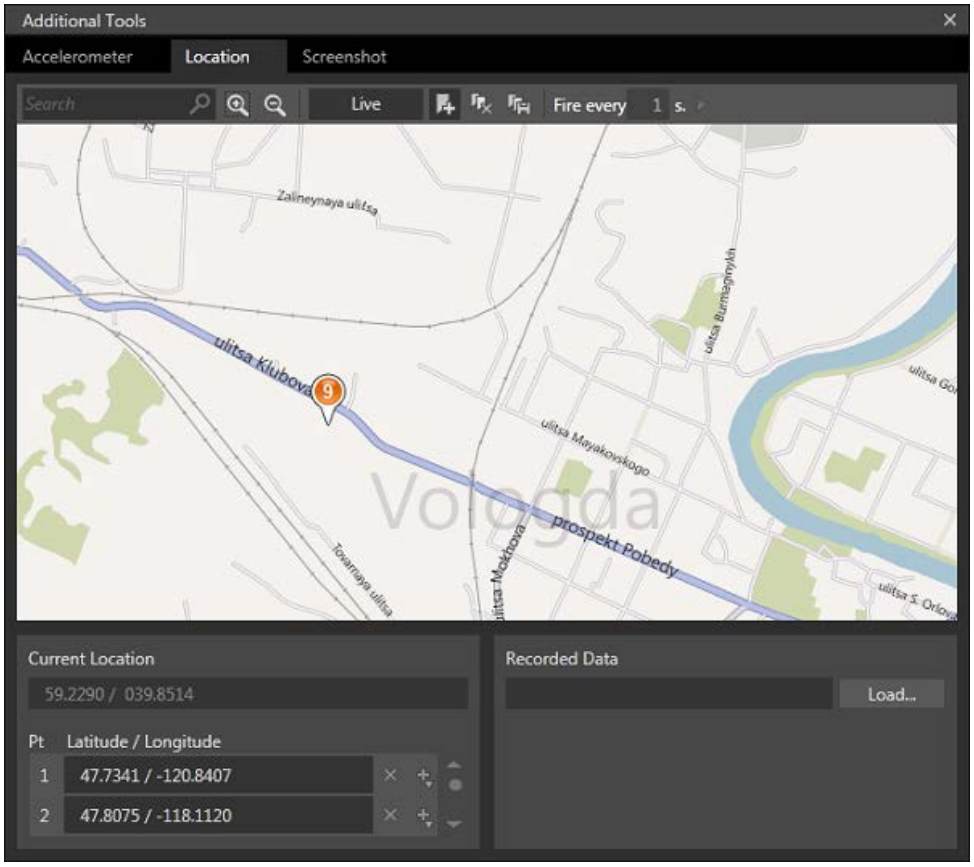
Создадим её в конструкторе класса:

myPushpin = new Pushpin();

В обработчике изменения позиции установим её на текущую позицию и добавим на карту, если её там нет:

```
myPushpin.Location = e.Position.Location;

if (!MyMap.Children.Contains(myPushpin)) MyMap.Children.Add(myPushpin);
```

Запустите приложение (F5) и воспользуйтесь возможностями эмулятора по эмуляции геолкационных данных, чтобы проверить работу программы. Увеличьте масштаб так, чтобы убедиться, что позиционирование и установка точки происходит правильно; также проверьте отображаемый статус сервисов геолокации.



Итоги и следующие шаги

Итак, мы познакомились с дополнительными возможностями платформы, которые может использовать разработчик мобильных приложений. Это были как программные возможности платформы, такие как задачи запуска и выбора, элементы управления Map и WebBrowser, так и аппаратные возможности работы с данными акселерометра и геолокации, доступные разработчику .

На следующем шаге мы познакомимся с возможностями платформы по локальному хранению данных и работе с веб.

Файлы для загрузки

Проект ExploreLaunchers
Проект ExploreChoosers
Проект ExploreMapControl
Проект ExploreMapControl с поддержкой геолокации
Проект ExploreMapControl с поддержкой акселерометра
Проект ExploreWebControl

Разработка под Windows Phone: Часть 4: Локальное хранение данных и работа с HTTP

Рейтинг

Возможность получать данные извне и сохранять их локально для дальнейшего использования или сохранять локально настройки приложение или результаты его работы – важная часть возможностей мобильной платформы.

На примере простого приложения мы научимся получать данные по протоколу HTTP и, при необходимости, сохранять их на устройстве.

Работа с HTTP и простой клиент для чтения RSS

Разработаем простой клиент для чтения RSS, который будет при запуске обращаться по определенному адресу, считывать RSS ленту и представлять её заголовки пользователю. Для простоты мы закодируем ссылку на RSS ленту прямо в код, в качестве примера ленты в нашем приложении будет использоваться RSS лента русского блога MSDN: http://blogs.msdn.com/b/rudevnews/rss.aspx

Наш клиент будет отображать только заголовки с возможностью перейти по ссылке на новость, так что разработку начнём со стандартного шаблона Windows Phone Application и назовём приложение SimpleRussianRSSReader.

Сначала научимся получать данные, а потом перейдем к их отображению.

Добавим в код константу – URL, указывающий на RSS:

const string RSS = "http://blogs.msdn.com/b/rudevnews/rss.aspx";

Теперь нам надо обратиться по указному адресу и скачать ленту.

Разработчику доступны два API: WebClient и HttpWebRequest. WebClient API позволяет удобно работать со GET/POST запросами, HttpWebRequest позволяет использоваться методы PUT/DELETE и даёт больше контроля разработчику над параметрами запроса. Поскольку нам надо просто скачать документ с веба, мы будем использовать WebClient.

Напишем простую функцию, в которой создадим экземпляр класса, зарегистрируемся на окончание процесса скачивания и запустим асинхронную процедуру скачивания документа:

```
1. private void LoadRSS()
2. {
3.      WebClient client = new WebClient();
4.      client.DownloadStringCompleted += new DownloadStringCompletedEventHandler(client_Download
   StringCompleted);
5.       client.DownloadStringAsync(new Uri(RSS));
6. }
```

Добавим глобальную строковую переменную в которой сохраним результат запроса:

```
string RSSString = "";
```

И в обработчике завершения скачивания, при отсутствии ошибок, присвоим ей значения результата:

```
1. void client_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
2. {
3.      if (e.Error == null)
4.      {
5.           RSSString = e.Result;
6.      }
7.  }
```

Чтобы проверить работоспособность кода на этом этапе, добавим элемент управления TextBlock на страницу приложения, заодно отредактировав его название, так что XAML код будет выглядеть следующим образом:

```
1. <!--TitlePanel contains the name of the application and page title-->
2.         <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
3.             <TextBlock x:Name="ApplicationTitle" Text="РУССКИЙ MSDN" Style="{StaticResource P
```


От новичка к эксперту

```
    honeTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="новости" Margin="9,-
    7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock Name="RSSText" ></TextBlock>
        </Grid>
```

В конструктор класса добавим вызов функции LoadRSS, а в обработчик загрузки, присвоение свойству Text элемента управления RSSText результата полученного из веб.

Запустите приложение (F5) и убедитесь, что мы получаем ответ от сервера.

Собственно мы научились основам работы с HTTP. Тот же самый запрос, реализованный через HttpWebRequest потребует значительных усилий, написания callback функции и определения множества дополнительных переменных.

Чтобы получить более красивый интерфейс отображения мы немного забежим вперёд. Настойчивый читатель сможет при желании разобраться с

Удалим TextBlock и добавим ListBox с шаблоном и привязкой к данным.

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <ListBox Name="RssList">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <TextBlock Text="{Binding pubDate}" FontSize="20" Foreground="Co
ral"/>
                            <TextBlock Text="{Binding title}" TextWrapping="Wrap" FontSize="
22"/>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
```

Теперь нужно создать класс, который будет содержать свойства pubDate и title. Список экземпляров этого класса мы получим разобрав при помощи LINQ полученный XML.

Добавьте в решение класс с именем PostMessage и определите в нем свойства pubDate и title типа

строка:

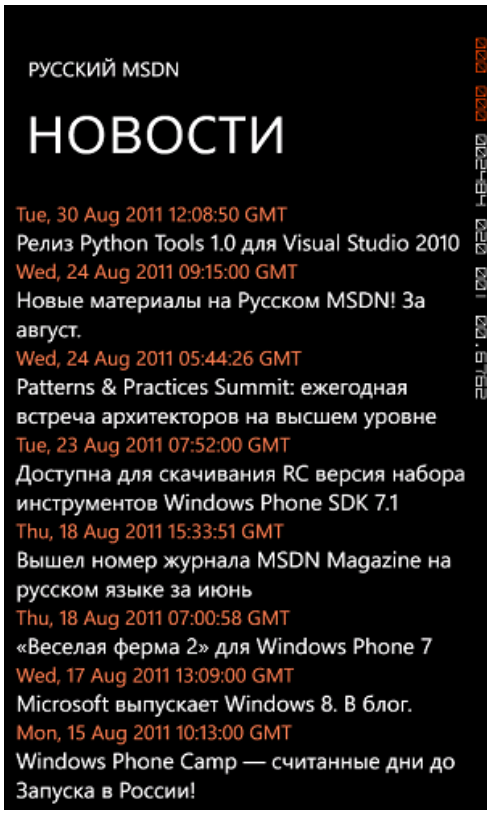
```
public class PostMessage
    {
        public string pubDate { get; set; }

        public string title { get; set; }
    }
```

Теперь добавьте в решение ссылку на библиотеку System.Xml.Linq и соответствующую директиву using в файл MainPage.xaml.cs

using System.Xml.Linq;
Теперь в обработчике результатов запроса мы можем обработать полученные результаты, проинициализировать список экземпляров класса и связать его с ListBox, чтобы отобразить в интерфейсе пользователя.

```
XElement twitterElements = XElement.Parse(e.Result);

                var postList =
                    from tweet in twitterElements.Descendants("item")
                    select new PostMessage
                    {
                        title = tweet.Element("title").Value,
                        pubDate = tweet.Element("pubDate").Value
                    };

                RssList.ItemsSource = postList;
```

Запустите приложение (F5) и проверьте, как работает наш простой клиент RSS.



Есть ещё много возможностей доработать данное приложение как в смысле дизайна визуального, так и программного. Например, можно отображать дату в соответствии с региональным настройками телефона, можно по щелчку по теме сообщения открывать возможность чтения тела сообщения в RSS – оставляем это для наших следующих частей или для самостоятельной работы читателей.

В качестве же последнего усовершенствования и закрепления материала по задачам запуска, модифицируем программу, чтобы при выборе темы открывался интернет браузер со страницей блога.

Добавим в класс PostMessage свойство link:

```
public class PostMessage
```

```
    {
        public string pubDate { get; set; }

        public string title { get; set; }

        public string link { get; set; }
    }
```

При разборе RSS добавим инициализацию этого поля:

```
var postList =
                    from tweet in twitterElements.Descendants("item")
                    select new PostMessage
                    {
                        title = tweet.Element("title").Value,
                        pubDate = tweet.Element("pubDate").Value,
                        link = tweet.Element("link").Value
                    };
```

В XAML файле MainPage назначим обработчик события SelectionChanged и напишем следующий код в этом обработчике:

```
private void RssList_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            WebBrowserTask webTask = new WebBrowserTask();
            webTask.Uri = new Uri(((PostMessage)(RssList.SelectedItem)).link);
            webTask.Show();
        }
```

Не забудьте добавить в секцию using следующую директиву:

using Microsoft.Phone.Tasks;
Запустите приложение (F5) и проверьте, как оно работает.

Локальное хранение данных

На платформе Windows Phone приложение может хранить данные тремя способами:

- **настройки**: данные сохраняются как пары ключ/значение, используется класс **IsolatedStorageSettings**;
- **файлы и папки** сохраняются на устройстве с использованием класса **IsolatedStorageFile**;
- **реляционные данные** сохраняются в локальной базе данных с использованием технологии LINQ в SQL.

Изолированное хранилище для настроек

Если говорить о настройках, то одна из основных возможностей изолированного хранилища для настроек - это его автоматическое сохранения при выходе пользователя из приложения.

Предназначено собственно для хранения настроек приложения, один экземпляр для одного приложения, создаётся при первом обращении.

Изолированное хранилище для файлов и папок

Чтобы разобраться, с основами работы с изолированным хранилищем, добавим в наше приложение возможность сохранять полученную RSS ленту между запусками и запрашивать сервер тогда, когда у нас лента не загружена или пользователь попросил обновить ленту.

Как обычно, сначала добавим в секцию using дополнительную директивы:

using System.IO.IsolatedStorage;
using System.IO;
Добавим функции сохранения и считывания файла. Для этого сначала определим константу с именем файла:

const string RSSFileName = "rss.xml";
Далее определим функции считывания и записи файла в строку из изолированного хранилища:

```
void SaveRSSToIsolatedStorage(string RSSText)
        {
            IsolatedStorageFile rssFileStorage = IsolatedStorageFile.GetUserStoreForApplicatio
n();
            IsolatedStorageFileStream rssFileStream = rssFileStorage.CreateFile(RSSFileName);

```

```
            StreamWriter sw = new StreamWriter(rssFileStream);
            sw.Write(RSSText);
            sw.Close();

            rssFileStream.Close();
        }


        string LoadRSSFromIsolatedStorage()
        {
            IsolatedStorageFile rssFileStorage = IsolatedStorageFile.GetUserStoreForApplicatio
    n();
            IsolatedStorageFileStream rssFileStream = rssFileStorage.OpenFile(RSSFileName, Sy
    stem.IO.FileMode.Open);

            StreamReader sr = new StreamReader(rssFileStream);
            string RSS = sr.ReadToEnd();
            sr.Close();
            rssFileStream.Close();

            return RSS;
        }
```

Для удобства добавления функционала также определим функцию, которая будет проверять наличие файла в изолированном хранилище.

```
bool IsRSSExist()
        {
            IsolatedStorageFile rssFileStorage = IsolatedStorageFile.GetUserStoreForApplicatio
    n();
            return rssFileStorage.FileExists(RSSFileName);
        }
```

Теперь выделим разбор полученного результата и связывание данных в отдельную функцию:

```
void ParseRSSAndBindData(string RSSText)
        {
            XElement twitterElements = XElement.Parse(RSSText);

            var postList =
                from tweet in twitterElements.Descendants("item")
                select new PostMessage
                {
                    title = tweet.Element("title").Value,
                    pubDate = tweet.Element("pubDate").Value,
                    link = tweet.Element("link").Value
                };

            RssList.ItemsSource = postList;
        }
```

И добавим сохранение полученного результата в обработчик завершения загрузки:

```
void client_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
        {
            if (e.Error == null)
            {
                RSSString = e.Result;

                ParseRSSAndBindData(RSSString);

                SaveRSSToIsolatedStorage(RSSString);
            }
        }
```

Осталось модифицировать функцию LoadRSS():

```
private void LoadRSS()
        {
            if (IsRSSExist())
            {
                RSSString = LoadRSSFromIsolatedStorage();
                ParseRSSAndBindData(RSSString);
```

```
            }
            else
            {
                RequestRSS();
            }
        }
```

Где RequestRSS() – это наша старая функция LoadRSS():

```
private void RequestRSS()
        {
            WebClient client = new WebClient();
            client.DownloadStringCompleted += new DownloadStringCompletedEventHandler(client_D
ownloadStringCompleted);
            client.DownloadStringAsync(new Uri(RSS));

        }
```

Запустите программу под отладчиком несколько раз и убедитесь, что после первого запуска программа сохраняет результат в изолированное хранилище и больше не обращается к вебу.

Самостоятельно измените интерфейс, добавив кнопку для обновления и дописав необходимый код.

Локальная база данных

А что, если в реальности мы не хотим замусоривать изолированное хранилище файлами, которые содержат лишнюю информацию, а также мы хотим выполнять анализ полученных данных, путем запросов к ним, тем или иным образом. Тут на помощь нам может прийти локальное хранилище реляционных данных.

Для того, чтобы им воспользоваться необходимо добавить в решение ссылку на библиотеку System.Data.Linq, а также добавить блок using следующие директивы:

```
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.ComponentModel;
using System.Collections.ObjectModel;
```

Далее необходимо определить классы, которое будут представлять сущности для хранения в локальной базе, отаттрибутировать их соответсвующим образом ([Table] и [Column] с параметрами) и реализовать 2 интерфейса INotifyPropertyChanged, INotifyPropertyChanging, чтобы минимизировать использование памяти.

Затем необходимо определить свой класс контекста данных, унаследованный от DataContext и определить в нем таблицы. Это создаст базовую инфраструктуру для использования локальной базы данных на устройстве.

Полный разбор работы с базой данных выходит за рамки этой статьи.

Для дальнейшего изучения работы с базой данных, можно посмотреть простой пошаговый пример создания приложения для работы с базой данных можно посмотреть здесь: http://msdn.microsoft.com/en-us/library/hh202876(v=VS.92).aspx

Более сложный пример приложения, работающего с локальной базой данных и сделанного в паттерне MVVM можно скачать здесь: http://go.microsoft.com/fwlink/?LinkId=219066 , краткое пояснение к проекту доступно по следующей ссылке http://msdn.microsoft.com/en-us/library/hh286405(v=VS.92).aspx

Итоги и следующие шаги

Итак, мы познакомились с тем, как получать данные из веб и сохранять их локально в файл. Также мы узнали, что существует локальная база данных, доступ к которой реализован аналогично работе с Entity Framework с поправкой на особенности платформы.

На следующем шаге мы познакомимся с жизненным циклом приложения, фоновыми сервисами и многозадачностью.

Файлы для загрузки

Проект SimpleRussianRSSReader
Проект SimpleRussianRSSReader с поддержкой Isolated Storage

Разработка под Windows Phone: Часть 5: Жизненный цикл приложения, фоновые сервисы и многозадачность

Рейтинг

Для того, чтобы писать хорошие приложения, которыми пользователю удобно пользоваться, необходимо понимать, что из себя представляет жизненный цикл приложения

Жизненный цикл и сохранение состояния приложения

В Windows Phone только одно приложение может быть активно/запущено в каждый момент времени. Когда приложение перестаёт быть активным, операционная система переводит его в спящее состояние (dormant). Если памяти устройства недостаточно для хорошей работы активного приложения, операционная система начинает завершать спящие (dormant) приложения. При этом, последними будет завершены приложения, которые запускались недавно.

Платформа Windows Phone предоставляет разработчику события, методы и объекты, которые он может использовать для необходимых действий на каждом жизненном цикле приложения.

Приложение позволяет разработчику обработать события Launching, Closing, Activated и Deactivated события. Поскольку пользователь покидает какую-то страницу или приходит на какую-то страницу приложения, с этими событиями связаны два доступных для переопределения метода страниц приложения: OnNavigatedTo и OnNavogatedFrom.

Для того, чтобы понять, как приложение и система взаимодействуют и какие возможности есть у разработчика, рассмотрим несколько сценариев жизненных циклов приложения.

Пользователь запускает приложение со стартовой страницы телефона или списка приложений, работает с приложением, затем нажимает кнопку **Back** телефона, находясь на **стартовой странице приложения**:

1. Приложение запускается, вызывается событие **Launching**
2. Загружается **стартовая страница приложения**, вызывается её метод **OnNavigatedTo**
3. Приложение работает.
4. Пользователь выходит из приложения, нажимая кнопку **Back**, находясь на **стартовой странице приложения**.
5. Вызывается метод **OnNavigatedFrom** стартовой страницы приложения
6. Вызывается событие **Closing** приложения.

Пользователь запускает приложение со стартовой страницы телефона или списка приложений, работает с приложением, затем нажимает кнопку **перехода к стартовой странице телефона**, находясь на **любой странице приложения**, а потом быстро возвращается обратно (операционная система не завершает приложение), используя длинное нажатие кнопки **Back** и выбирая страницу приложения, с которой он ушёл.

1. Приложение запускается, вызывается событие **Launching**
2. Загружается **стартовая страница приложения**, вызывается её метод **OnNavigatedTo**
3. Приложение работает.
4. Пользователь выходит из приложения, нажимая кнопку **Back**, находясь на **стартовой странице приложения**.
5. Вызывается метод **OnNavigatedFrom** стартовой страницы приложения
6. Вызывается событие **Deactivated** приложения.
7. Приложение переходит в спящее (dormant) состояние.
8. Пользователь не интенсивно пользуется телефоном, так что выгрузки приложения из спящего состояния не происходит.
9. Пользователь возвращается в приложение, используя длинное нажатие кнопки **Back** и выбирая страницу приложения, с которой он ушёл.
10. Вызывается событие **Activated** приложения.
11. Загружается страница приложения с которой ушёл пользователь и вызывается её метод **OnNavigatedTo**
12. Приложение работает.

От новичка к эксперту

Что произойдёт, если операционной системе потребуется больше памяти, когда приложение находится в памяти в спящем состоянии?

Тогда приложение завершит свою работу, но сохранит состояние стека навигации, а также состояние словарей состояния на уровне приложения (**PhoneApplicationService.State**) и на уровне страницы (**PhoneApplicationPage.State**) – перейдёт в состояние Tombstoned. Обратите внимание, чтоы выход из которого в разрезе возникающих событий и вызываемых методов не отличается от выхода из спящего (dormant) состояния, но при этом, поскольку приложение будет выгружено из памяти, необходимо позаботиться о сохранении данных для восстановления состояния приложения. Узнать об это разработчик может, проверив свойство IsApplicationInstancePreserved у ActivatedEventArgs. В случае true – это восстановление из спящего состояния, в случае же false – из Tombstone состояния.

Одновременно система сохраняет состояние Tombstoned только для пяти приложений. Если пользователь не возвращается к приложению, данные удаляются, и приложение будет запускаться с событием **Launching**.

Подробнее о жизненом цикле приложения можно прочитать по следующей ссылке:
http://msdn.microsoft.com/en-us/library/ff769557(v=VS.92).aspx

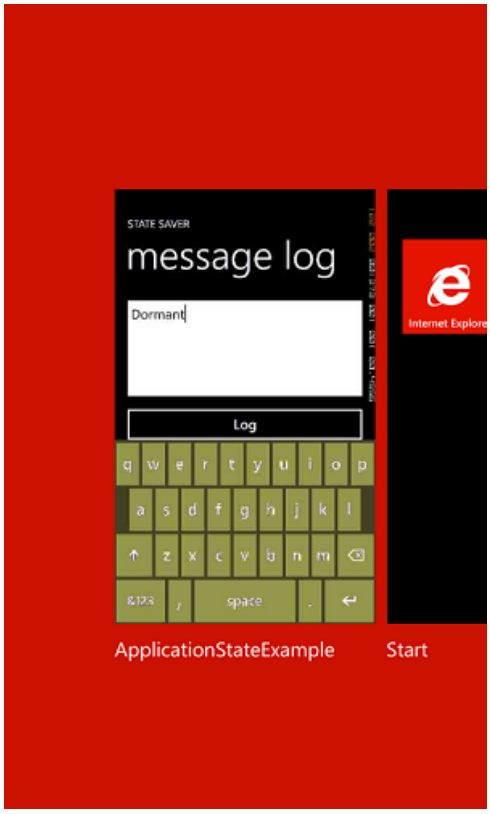
Давайте теперь на практике попробуем разобраться с сохранением состояния приложения.

Создадим новое приложение на базе стандартного шаблона Windows Phone Application и назовём приложение ApplicationStateExample.

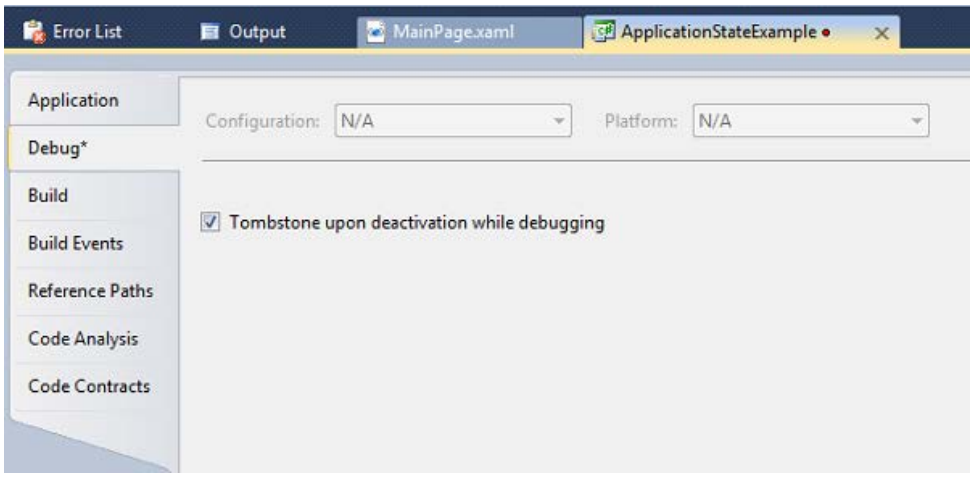
Исправьте код XAML страницы приложения на следующий:

```
<!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="STATE SAVER" Style="{StaticResource P
honeTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="message log" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <TextBox Name="Message" Height="200" TextWrapping="Wrap"></TextBox>
                <Button Name="Log" Height="80" Content="Log"></Button>
            </StackPanel>
        </Grid>
```

Соберите приложение и запустите его (F5). Наберите что-нибудь в поле ввода, а затем перейдите на стартовую страницу, нажав среднюю кнопку на эмуляторе. Потом нажмите и удержите кнопку **Back** отобразится список доступных приложений, выберите наше приложение. Обратите внимание, что состояние страницы сохранилось, т.к. приложение было в спящем (dormant) состоянии без какого либо участия с нашей стороны.



Завершите работу приложения, перейдите в настройки отладки и установите настройку Tombstone upon deactivation while debugging:



Соберите приложение и запустите его (F5). Наберите что-нибудь в поле ввода, а затем перейдите на стартовую страницу, нажав среднюю кнопку на эмуляторе. Потом нажмите и удержите кнопку **Back** отобразится список доступных приложений, выберите наше приложение. Обратите внимание, что состояние страницы не сохранилось, т.к. приложение было выгружено и перешло в состояние Tombstone, так что состояние страницы и приложения было утеряно.

Добавим в код страницы MainPage.xaml.cs функции OnNavigatedFrom и OnNavigatedTo, сохранив текст из поля ввода в словарь состояний:

```
protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);

            this.State.Add("text", Message.Text);
        }

        protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e
    )
        {
            base.OnNavigatedTo(e);

            if (this.State.ContainsKey("text"))
```

```
            {
                Message.Text = (string)this.State["text"];
            }
        }
```

Соберите приложение и запустите его (F5). Наберите что-нибудь в поле ввода, а затем перейдите на стартовую страницу, нажав среднюю кнопку на эмуляторе. Потом нажмите и удержите кнопку **Back** отобразится список доступных приложений, выберите наше приложение. Обратите внимание, что состояние страницы сохранилось, несмотря на то, что приложение было выгружено и перешло в состояние Tombstone, т.к. мы сохранили состояние страницы в специальном словаре для сохранения состояния страницы.

Это не совсем правильное поведение кода, т.к. в случае, если мы восстанавливаемся из спящего состояния, нам не стоит модифицировать состояние страниц.

Давайте добавим логическую переменную, чтобы определить, вызвался конструктор (это значить, что, мы либо стартуем первый раз, либо восстанавливается из tomnstone состояния) и модифицируем код соответствующим образом:

```
bool isNewlyCreatedPage = false;

        // Constructor
        public MainPage()
        {
            InitializeComponent();

            isNewlyCreatedPage = true;
        }
```

И

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);

            if (this.State.ContainsKey("text") && isNewlyCreatedPage)
            {
                Message.Text = (string)this.State["text"];
            }

            isNewlyCreatedPage = false;
        }
```

Запустите приложение (F5) и проверьте, как оно работает.

Снимите флажок настройки отладки Tombstone upon deactivation while debugging и проверьте, что приложение не восстанавливает состояние в случае выхода из спящего (dormant) состояния.

Добавим в приложение код, чтобы попробовать сохранить состояние приложения.

Добавим TextBlock и обработчик Click в XAML код страницы:

```
<StackPanel>
                <TextBox Name="Message" Height="200" TextWrapping="Wrap"></TextBox>
                <Button Name="Log" Height="80" Content="Log" Click="Log_Click"></Button>
                <TextBlock Name="AppState" Height="80"></TextBlock>
            </StackPanel>
```

В файл App.xaml.cs добавим публичное поле AppState:

public string AppState = "";

Модифицируем код файда MainPage.xaml.cs, чтобы при нажатии на кнопку сообщение из TextBox записывалось в AppState и отображалось в TextBlock, а при восстановление приложения AppState отображалось в TextBlock:

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);

            if (this.State.ContainsKey("text") && isNewlyCreatedPage)
            {
```

```
            Message.Text = (string)this.State["text"];

            App myApp = App.Current as App;
            AppState.Text = myApp.AppState;
        }

        isNewlyCreatedPage = false;
    }

    private void Log_Click(object sender, RoutedEventArgs e)
    {
        AppState.Text = Message.Text;

        App myApp = App.Current as App;

        myApp.AppState = AppState.Text;
    }
```

Теперь необходимо добавить код сохранения/восстановления в App.xaml.cs:

```
// Code to execute when the application is activated (brought to foreground)
    // This code will not execute when the application is first launched
    private void Application_Activated(object sender, ActivatedEventArgs e)
    {
        if (!e.IsApplicationInstancePreserved)
        {
            AppState = (string)PhoneApplicationService.Current.State["appState"];

        }
    }

    // Code to execute when the application is deactivated (sent to background)
    // This code will not execute when the application is closing
    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
    {
        PhoneApplicationService.Current.State.Add("appState", AppState);
    }
```

Перейдите в настройки отладки и установите флажок Tombstone upon deactivation while debugging, а затем запустите приложение (F5) и проверьте, что состояние приложения сохраняется.

Обратите внимание, что когда приложение вызывает задачи выбора и запуска, также возникает события Deactivated.

Многозадачность и фоновые сервисы

Несмотря на то, что в Windows Phone только одно приложение может быть активным, оно может воспользоваться специальными возможностями платформы, чтобы выполнять определенные задачи, даже не являясь активным.

Специальные возможности платформы включают в себя возможность создания фоновых сервисов, запускаемые по расписанию, возможность проигрывания музыки и загрузки/выгрузки файлов в фоновом режиме, а также регистрацию оповещения (Alarms) и напоминания (Reminders). Об оповещениях и напоминаниях мы поговорим позже, сейчас кратко остановимся на фоновых сервисах.

Фоновые сервисы, запускаемые по расписанию

Можно создавать два типа сервисов: периодический (PeriodicTask) и интенсивный (ResourceIntensiveTask). Периодический сервис запускается регулярно на небольшое время, интенсивный – запускается на относительно длительное время, но только если выполнятся определённые требования, относящиеся к состоянию телефона. API создания сервисов доступен в пространстве имён Microsoft.Phone.Scheduler.

Есть определенные ограничения, связанные в целом с сервисами, запускаемыми по расписанию, и с каждым типом сервисов в отдельности:

- приложение может иметь только один фоновый сервис, запускающийся по расписанию, но он может быть зарегистрирован и как периодический и как интенсивный;
- не весь API доступный на устройстве доступен в сервисах;
- сервис необходимо инициализировать в активном приложении и он будет сохраняться между перезагрузками телефона;
- у пользователя есть возможность контролировать его работу в настройках телефона;
- максимальное время работы сервиса – 14 дней (после этого можно инициализировать заново);
- если агент завершит свою работу два раза подряд из-за превышения ограничений на память или из-за необработанного исключения, он будет удалён из расписания запуска;

-
- количество сервисов на устройстве ограничено и зависит от конфигурации устройства.

Ниже приведена сравнительная таблица ограничений на периодический и интенсивный сервисы:

	Периодический	**Интенсивный**
Интервал запуска	Обычно, 30 минут, но в зависимости от состояния телефона (батарейка, запущенные процессы) может сдвигаться +/- 10 минут	Телефон подключён к внешнему источнику питания Доступ в сеть не через сотовую связь Заряд батареи не менее 90% Нет активного звонка Телефон залочен
Время работы	Порядка 25 секунд	Порядка 10 минут
Ресурсы	Меньше 6 Мб памяти, меньше 10% ресурсов процессора	Меньше 6 Мб памяти

Как мы уже говорили выше, не весь API доступен для использования в сервисах. Подробно со списком API можно познакомиться в документации MSDN: http://msdn.microsoft.com/en-us/library/hh202962(v=VS.92).aspx Ниже небольшая таблица, чтобы можно было получить представление о том, какой API доступен, а какой нет:

API доступен	**API запрещён**
Tiles	Работа с UI
Toast	XNA
Location	Микрофон и камера
Network	Сенсоры
Isolated Storage	Проигрывание аудио
Sockets	Скачивание файлов
Большинство Silverlight API	Регистрация новых сервисов

Теперь у нас есть неплохая теоретическая подготовка по фоновым сервисам, которые запускаются по расписанию.

Создайте новое приложение из шаблона Windows Phone Application, назовём его BackgroundAgentExample. В Solution Explorer, щелкнув правой кнопкой мыши по решению, в отобразившемся меню выберите Add, затем Add, затем New Project. Выберите тип проекта Windows Phone Scheduled Task Agent и назовите его ToastAgent.

Перейдите к проекту ToastAgent, двойным щелчком перейдите к редактированию файла ScheduledAgent.cs.

Добавьте в блок using код:

using Microsoft.Phone.Shell;
А в процедуру OnInvoke, которая вызывается при запуске сервиса, добавьте код отображения toast сообщения:

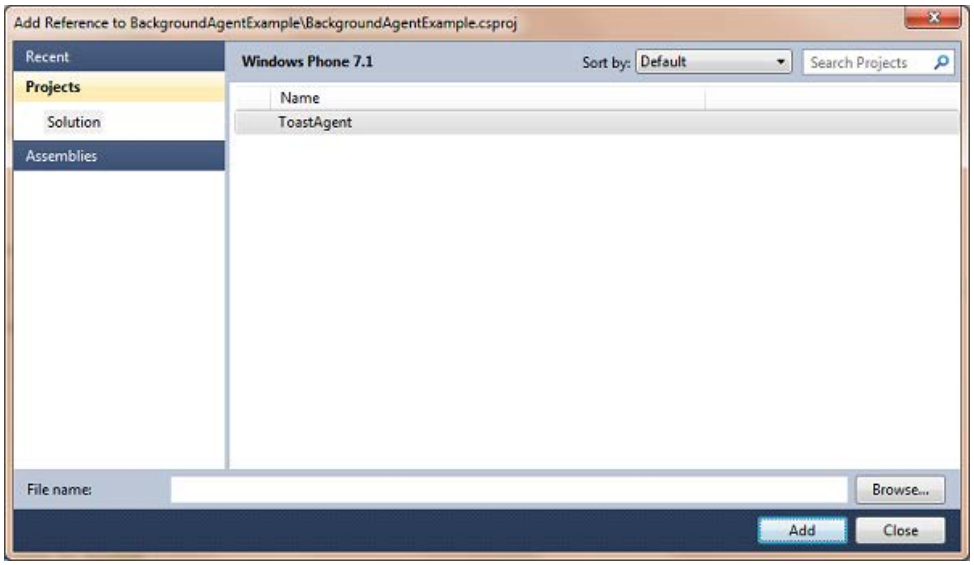
```
1.  protected override void OnInvoke(ScheduledTask task)
2.          {
3.              ShellToast toast = new ShellToast();
4.              toast.Title = "Toast Agent";
5.              toast.Content = "Сообщение от фонового сервиса";
6.              toast.Show();
7.
8.  #if DEBUG
9.              ScheduledActionService.LaunchForTest(task.Name, System.TimeSpan.FromSeconds(10));
10. #endif
11.
12.             NotifyComplete();
```

```
13.         }
```

Обратите внимание на код внутри директив условной компиляции. Он позволяет не ждать 30 минут запуска сервиса, а запустить его через 10 секунд после его последнего запуска.

Вернёмся теперь к основному проекту.

Сначала добавим в него ссылку на проект ToastAgent. Для этого щелкните левой кнопкой мыши по папке References, в выпадающем меню выберите Add Refernce, в отобразившемся диалоговом окне выберите слева Projects, Solution, в центральной части ToastAgent и нажмите кнопку Add, а потом Close.



Перейдём к редактированию кода MainPage.xaml. Добавим кнопку, чтобы запустить наш периодический сервис и добавим обработчик события Click:

```
1. <!--ContentPanel - place additional content here-->
2.         <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
3.             <Button Content="Start Toast" Height="150" Name="StartAgent" Click="StartAgent_C
   lick" />
4.         </Grid>
```

Добавим в блок using директиву:

using Microsoft.Phone.Scheduler;

и определим константу с именем сервиса в классе MainPage:

const string ToastAgentName = "Agent-Toast";

Отредактируем обработчик события нажатия кнопки так, чтобы запускался сервис ToastAgent.

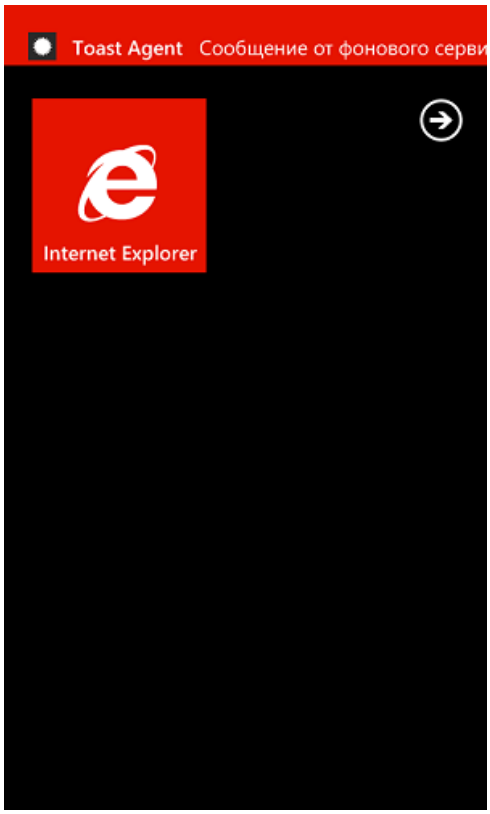
```
1. private void StartAgent_Click(object sender, RoutedEventArgs e)
2.         {
3.             PeriodicTask myPeriodicTask = ScheduledActionService.Find(ToastAgentName) as Peri
   odicTask;
4.
5.             if (myPeriodicTask != null)
6.             {
7.                 try
8.                 {
9.                     ScheduledActionService.Remove(ToastAgentName);
10.                }
11.                catch (Exception ex)
12.                {
13.                    MessageBox.Show("Невозможно удалить ранее созданный сервис:"+ex.Message);
14.                }
15.            }
16.
17.            myPeriodicTask = new PeriodicTask(ToastAgentName);
18.           myPeriodicTask.Description = "Agent-Toast";
19.
20.
21.            try
22.            {
```

```
23.                 ScheduledActionService.Add(myPeriodicTask);
24.
25.
26.
27. #if DEBUG
28.                 ScheduledActionService.LaunchForTest(ToastAgentName, TimeSpan.FromSeconds(10))
    ;
29. #endif
30.             }
31.             catch (Exception ex)
32.             {
33.                 MessageBox.Show("Невозможно создать сервис:" + ex.Message);
34.             }
35.
36.         }
```

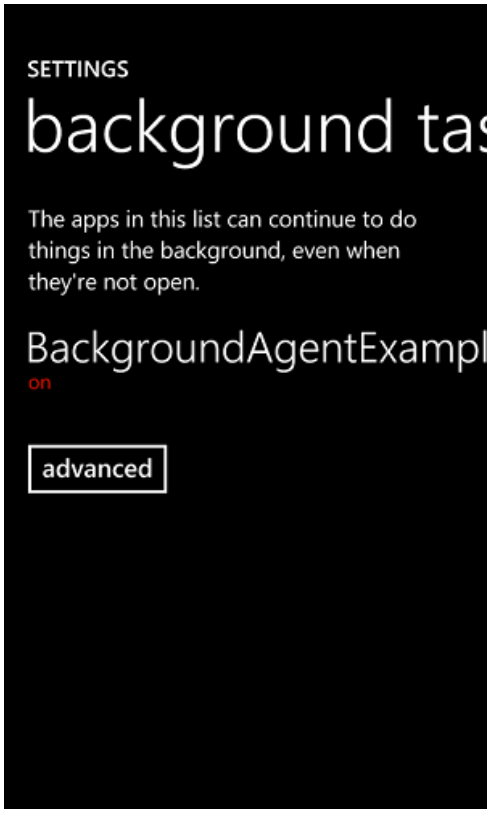
Поле описание сервиса является обязательным, без него не удастся добавить сервис.

Обратите внимание, что и здесь присутствует код, который в случае отладки вызывает сервис через 10 секунд.

Запустите приложение (F5) и протестируйте его работу: запустите приложение, нажмите кнопку Start Toast, выйдите из приложения, дождитесь появления сообщения от сервиса.



Перейдите в настройки приложений, в панель background tasks и проверьте, что наш сервис там действительно зарегистрирован



В качестве самостоятельного упражнения, попробуйте обработать вариант, когда сервис отключен пользователем, не используя исключений.

Фоновая загрузка/выгрузка файлов

Для фоновой загрузки/выгрузки файлов существует отдельный API, находящийся в пространстве имён Microsoft.Phone.BackgroundTransfer.

Файлы можно загружать/выгружать из/в изолированное хранилище (Isolated Storage) приложения, при этом процесс будет продолжаться, даже если приложение не запущено. Если приложение запущено, но оно может отлеживать состояние и отображать его статус фоновой загрузки файлов. Пока поддерживаются GET/POST HTTP/HTTPS запросы для скачивания файлов и POST HTTP/HTTPS запросы для загрузки файла на сервер.

Также существуют ограничения на размеры файлов, которые можно скачать/загрузить на сервер. На сервер можно загрузить файл не больше 5 Мб, скачать файл размером до 20 Мб можно по сотовой связи, файлы размером до 100 Мб – по беспроводной (WiFi) (если нет подключения к источнику питания).

Каждое приложение может иметь до 5 одновременных запросов на фоновую загрузку/выгрузку файлов. Обратите внимание, что при завершении загрузки/выгрузки запросы не удаляются – это необходимо сделать вручную. При этом, всего на устройстве может быть до 500 запросов на фоновую загрузку/выгрузку файлов.

Теперь мы знаем достаточно, чтобы добавить в ранее созданное приложение SimpeRussianRSSReader поддержу фоновой загрузки RSS.

Откройте последнюю версию приложения. Если у нас был получен RSS и сохранён в изолированное хранилище, обновление RSS автоматически не происходит.

Добавим запуск запроса на фоновое скачивание нового RSS файла, если у нас уже есть ранее скачанный.

Для начала в конструктор страницы добавим проверку, что в изолированном хранилище приложения присутствует специальная директория в которую можно скачивать файлы используя фоновые сервисы: /shared/transfers

```
1. using (IsolatedStorageFile rssStore = IsolatedStorageFile.GetUserStoreForApplication())
2.             {
3.                 if (!rssStore.DirectoryExists("/shared/transfers"))
4.                 {
5.                     rssStore.CreateDirectory("/shared/transfers");
6.                 }
7.             }
```

Эта директория создаётся системой для приложения, но может быть им удалена. Только файлы из этой директории можно загружать на сервер и только в эту директорию их можно выгружать.

В код функции LoadRSS, добавим код создания запроса на фоновую загрузку:

```
Uri transferUri = new Uri(RSS);

                BackgroundTransferRequest transferRequest = new BackgroundTransferRequest(tra
nsferUri);

                transferRequest.Method = "GET";


                Uri downloadUri = new Uri("shared/transfers/" + RSSFileName, UriKind.Relative
OrAbsolute);
                transferRequest.DownloadLocation = downloadUri;

                transferRequest.TransferStatusChanged += new EventHandler<BackgroundTransferEv
entArgs>(transferRequest_TransferStatusChanged);

                try
                {
                    BackgroundTransferService.Add(transferRequest);
                }
                catch (Exception ex)
                {

                    MessageBox.Show("Невозможно запустить фоновую загрузку:" + ex.Message);
                }
```

Мы берем URL по которому отдаётся RSS, создаём URI, указываем имя файла в директории /shared/transfers в которую мы хотим скачать RSS методом GET, создаём загрузку, регистрируем обработчик события изменения статуса загрузки и добавляем запрос на загрузку с систему.

Теперь, добавим в обработчик изменения статуса код, чтобы удалить запрос на загрузку, скопировать файл в корневую директорию и обновить пользовательский интерфейс:

```
void transferRequest_TransferStatusChanged(object sender, BackgroundTransferEventArgs e)
        {
            //если скачивание прошло успешно - копируем файл и обновляем UI
            if (e.Request.TransferStatus == TransferStatus.Completed)
            {
                try
                {
                    BackgroundTransferService.Remove(e.Request);
                }
                catch (Exception ex)
                {

                    MessageBox.Show("Невозможно удалить завершённую фоновую загрузку:" + ex.M
essage);
                }

                using (IsolatedStorageFile rssStore = IsolatedStorageFile.GetUserStoreForAppl
ication())
                {
                    if (rssStore.FileExists("/shared/transfers/" + RSSFileName))
                    {
                        rssStore.CopyFile("/shared/transfers/" + RSSFileName, RSSFileName, tr
ue);

                        RSSString = LoadRSSFromIsolatedStorage();
                        ParseRSSAndBindData(RSSString);
                    }
                }

            }
        }
```

Чтобы протестировать приложение, добавим ещё одно закоментированное значение константы, которая используется в качестве URL для скачивания RSS:

```
//const string RSS = "http://blogs.msdn.com/b/rustudents/rss.aspx";
```

Закройте эмулятор, чтобы удалились ранее загруженные в него данные и запустите приложение (F5) в первый раз, чтобы RSS скатался в изолированное хранилище и отобразился в интерфейсе пользователя. После того, как отобразится список заголовков RSS, завершите работу приложения в Visual Studio (не закрывайте эмулятор).

Чтобы увидеть, что произошла фоновая загрузка, закомментируем оригиналньый URL и раскоментируем другой:

```
1. //const string RSS = "http://blogs.msdn.com/b/rudevnews/rss.aspx";
2. const string RSS = "http://blogs.msdn.com/b/rustudents/rss.aspx";
```

Запустите приложение (F5) подождите некоторое время – заголовки RSS обновятся соответствующим образом.

Фоновое проигрывание музыки

Также существует отдельный API для проигрывания локальной и потоковой музыки, находящийся в пространстве имён Microsoft.Phone.BackgroundAudio. Создание приложений, которые используют эту возможность аналогично созданию приложений для фоновых сервисов запускаемых по расписанию, с поправкой на специфику проигрывания аудио.

В поставке средств разработки находятся два шаблона проектов Windows Audio Playback Agent и Windows Audio Streaming Agent для создания сервисов фонового проигрывания локальной и потоковой музыки соответственно.

Особенностью данных сервисов является их тесная интеграция с платформой Windows Phone.

Подробнее с архитектурой сервисов фонового проигрывания музыки можно познакомиться в документации MSDN: http://msdn.microsoft.com/ru-ru/library/hh394039(v=VS.92).aspx

По ссылке http://msdn.microsoft.com/ru-ru/library/hh202978(v=VS.92).aspx доступно пошаговое руководство по созданию простого приложения, проигрывающего локальные аудиофайлы в фоновом режиме.

Итоги и следующие шаги

В этой статье мы познакомились с жизненным циклом приложения, фоновыми сервисами и многозадачностью, в следующей части мы познакомимся с уведомлениями и напоминаниями, а также с Live Tiles.

Файлы для загрузки

Проект ApplicationStateExample
Проект BackgroundAgentExample
Проект SimpleRussianRSSReader с Background File Transfer

Разработка под Windows Phone: Часть 6: Оповещения, Live Tiles и Push Notification

Рейтинг

Так ли нужна одновременная работа нескольких приложений в фоновом режиме? На самом деле, в большинстве случаев, нам нужно, чтобы приложение имело возможность оповестить пользователя о том, что что-то изменилось или необходимы какие-то действия с его стороны. И в этой части серии статей мы познакомимся с возможностями системы, которые позволяют нам оповещать пользователя, напоминать ему или даже предоставлять саму востребованную информацию без необходимости запускать приложение!

Предупреждения (Alarm) и напоминания (Reminder)

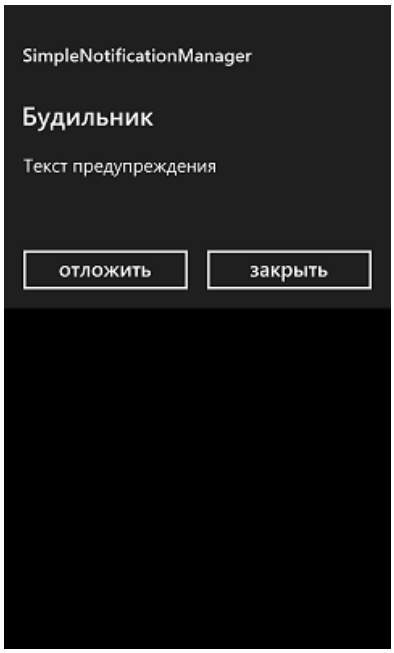
Приложения в Windows Phone могут использовать оповещения двух типов: предупреждения (Alarms) и напоминания (Reminders), которые будут отображаться пользователю в виде диалоговых окон, по расписанию. Интерфейс оповещений и напоминаний соответствует тому, который используют системные приложения. Это позволяет сторонним приложениям общаться с пользователем на знакомом ему языке системного интерфейса.

Объекты Alarm и Reminder наследуются от класса ScheduledNotification и имеют достаточно много общего, однако, есть и определенный отличия. Классический вариант оповещения Alarm – это будильник, а оповещения Reminder – напоминание о событии в календаре.

Обратите внимание, что каждое приложение может зарегистрировать до 50 оповещений, а точность отображения пользователю – около минуты.

Давайте кратко рассмотрим основные параметры оповещений.

Предупреждения (Alarm)



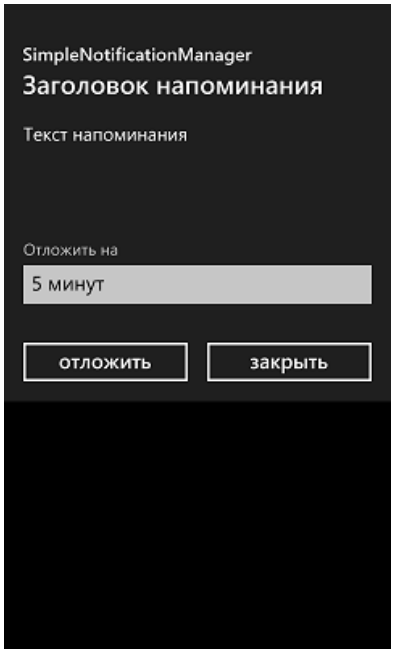
Предупреждение (Alarm) отображается пользователю в виде диалогового окна с 2-мя кнопками отложить (snooze) и закрыть (dismiss), а также тремя текстовыми блоками. Верхний блок – название приложения, которое зарегистрировало предупреждение. Далее, текстовый блок, который отображается всегда и не может быть изменён с надписью Будильник (Alarm) и, самый последний блок – текст, который был указан при создании предупреждения.

Предупреждение (Alarm) позволяет указать звуковой файл, который будет проигран при отображении его пользователю, он проигрывается с нарастающей громкостью.

Если пользователь коснётся любой области оповещения, кроме кнопок, будет загружено основное приложение и отобразится его начальная страница, так, как будто пользователь запустил его из списка программ.

Напоминание (Reminder)

От новичка к эксперту



Напоминание (Reminder) отображается пользователю в виде диалогового окна с 2-мя кнопками, выпадающим списком для выбора, на сколько отложить оповещение и тремя текстовыми блоками.

Верхний блок – название приложения, которое зарегистрировало предупреждение. Далее, текстовый блок, который отображает заголовок оповещения, который был указан при его создании, и, самый последний блок – текст – содержание напоминания, указанный при создании.

Если пользователь коснётся любой области оповещения, кроме кнопок, будет загружено основное приложение, при этом, разработчик, при создании напоминания может указать URI страницы с параметрами, на которые перейдёт пользователь.

В отличие от предупреждения (Alarm) будет вегда использоваться звук для оповещения установленный в настройках устройства.

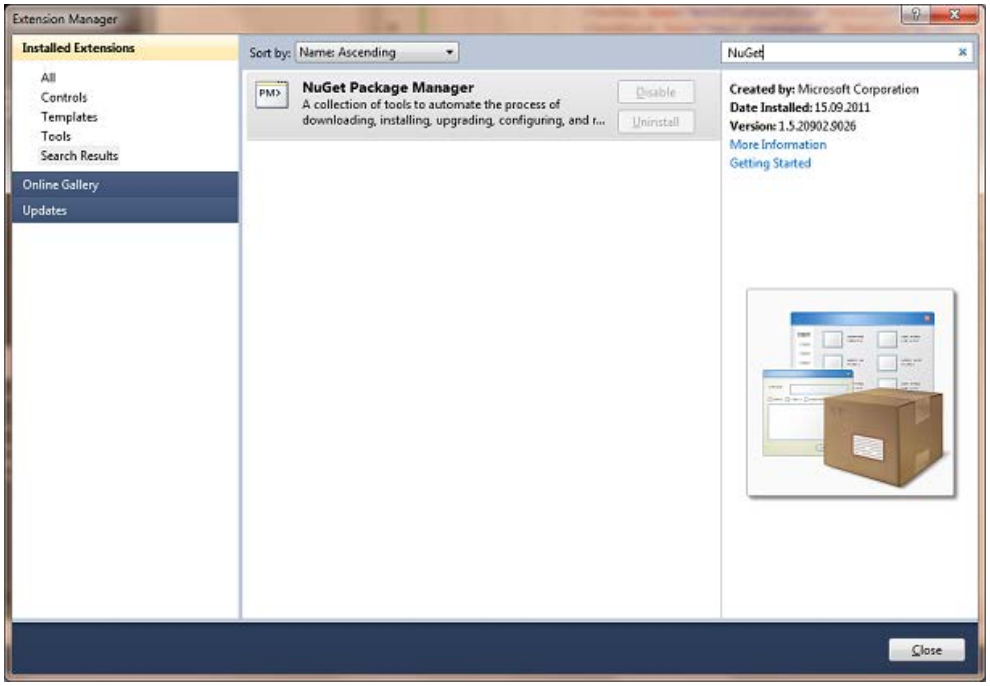
Пишем менеджер оповещений

Давайте теперь на практике попробуем разобраться с оповещениями.

Создадим новое приложение на базе стандартного шаблона Windows Phone Application и назовём приложение SimpleNotificationManager.

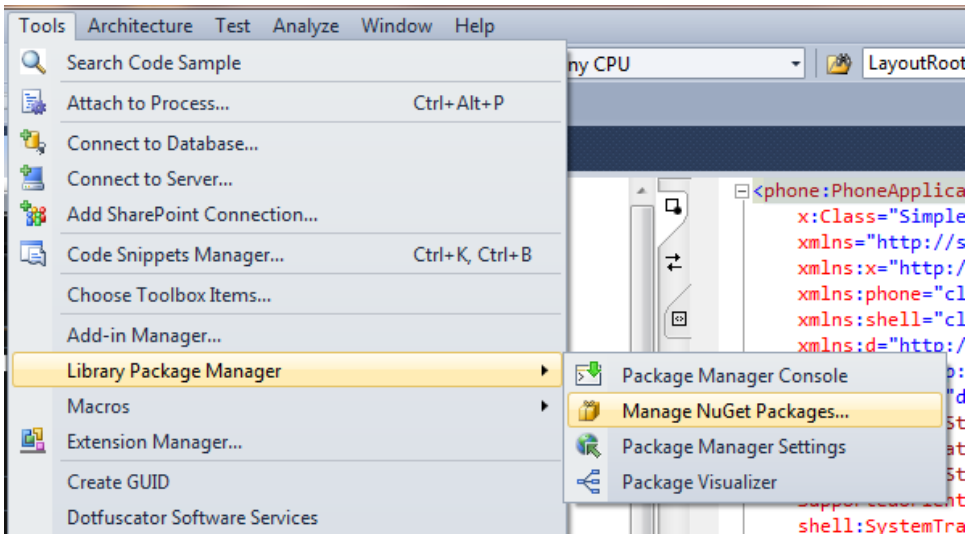
После того, как проект приложения будет создан, добавьте в приложение поддержку Silverlight for Windows Phone Toolkit.

Убедитесь, что у вас установлен NuGet Package Manager (Tools -> Extension Manager ...)

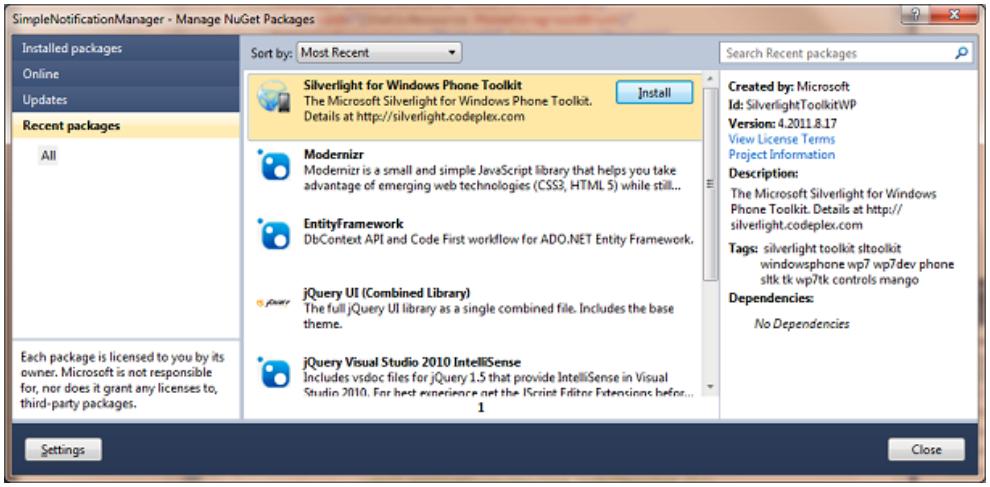


При необходимости – установите.

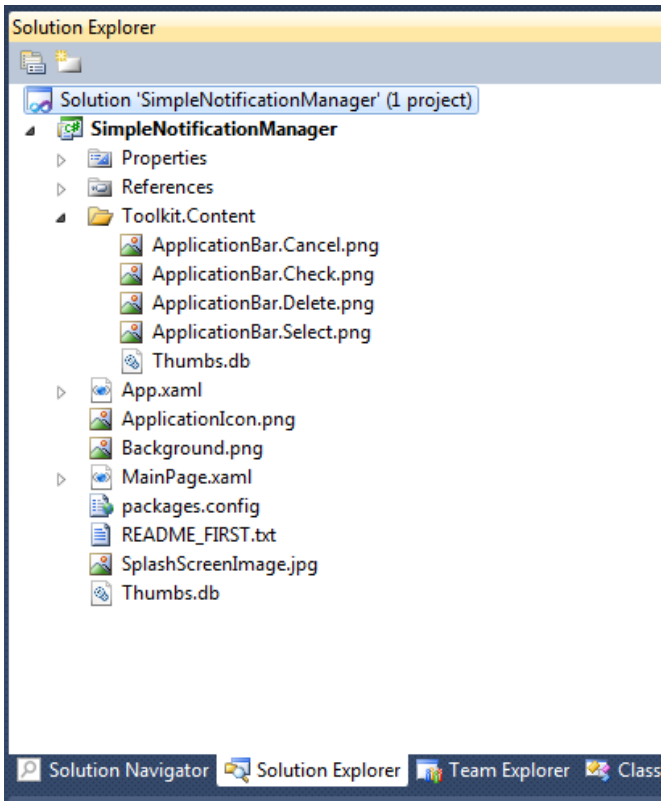
После этого в меню Visual Studio выберите, Tools -> Library Package Manager -> Manage NuGet Packages ...



Откроется графическая утилита установки пактов NuGet. Найдите и установите пакет Silverlight for Windows Phone Toolkit.

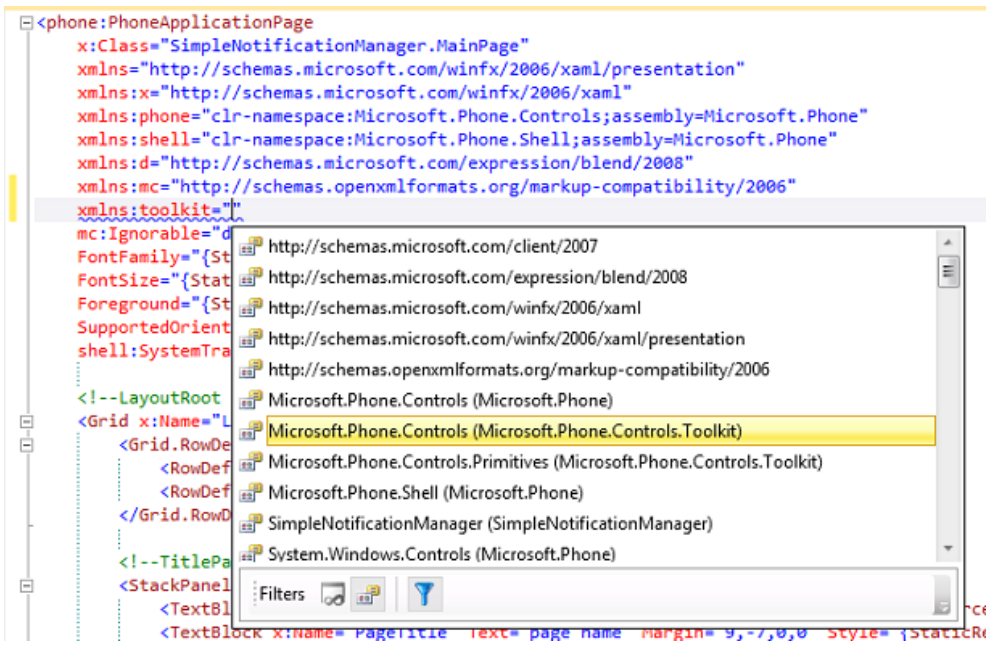


После этого, ваш проект будет выглядеть следующим образом.



Перед тем, как продолжить, для всех картинок в папке Toolkit.Content установите тип сборки (Build Action) в Content (по умолчанию стоит Resource).

Итак, Silverlight for Windows Phone Toolkit подключён к проекту. Давайте добавим, как мы это уже делали ранее, пространство имён toolkit в XAML файл MainPage.xaml, чтобы использовать элементы управления из Silverlight for Windows Phone Toolkit на нашей главной странице:



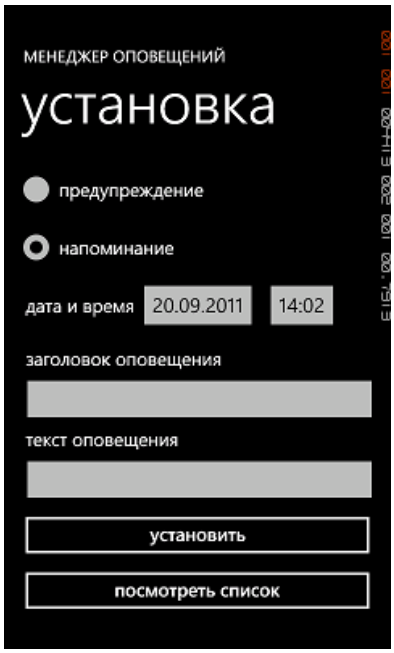
Мы добавили следующую строку:

```
1.  xmlns:toolkit="clr-
    namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
```

И в результате XAML будет выглядеть следующим образом:

```
<phone:PhoneApplicationPage
    x:Class="SimpleNotificationManager.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">
```

Теперь нужно сделать интерфейс создания оповещений. Нужен выбор типа, выбор времени оповещения, заголовка (в случае напоминания) и содержания. Я сделал простой интерфейс на базе комбинации элементов управления StackPanel:



При этом, XAML код страницы MainPage.xaml (внутри тега phone:PhoneApplicationPage) выглядит следующим образом:

```
<!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="МЕНЕДЖЕР ОПОВЕЩЕНИЙ" Style="{StaticRes
ource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="установка" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <RadioButton Content="предупреждение" GroupName="NotificationType" Name="Alarm
Type" IsChecked="True" />
                <RadioButton Content="напоминание" GroupName="NotificationType" Name="Reminder
Type" />
                <StackPanel Orientation="Horizontal" Margin="15,0,0,0">
                    <TextBlock Text="дата и время"  FontSize="22.667" VerticalAlignment="Cent
er"/>
                    <toolkit:DatePicker Name="NotificationDate"/>
                    <toolkit:TimePicker Name="NotificationTime"/>
```

```
                </StackPanel>
                <StackPanel  Margin="15,15,0,0">
                    <TextBlock Name="NotificationTitleCaption" Text="заголовок оповещения"  F
ontSize="22.667" VerticalAlignment="Center" Visibility="Collapsed"/>
                    <TextBox Name="NotificationTitle" FontSize="22.667" Margin="-
10,0,0,0"  Visibility="Collapsed"/>
                    <TextBlock Text="текст оповещения"  FontSize="22.667" VerticalAlignment="
Center"/>
                    <TextBox Name="NotificationText" FontSize="22.667" Margin="-10,0,0,0"/>
                </StackPanel>
                <Button Content="установить" Name="SetNotification" FontSize="22.667" Width="
450" />
                <Button Content="посмотреть список" Name="CheckNotification" FontSize="22.667
" Width="450" />
                </StackPanel>
        </Grid>
    </Grid>
```

Добавьте в проект ещё одну страницу и назовите её NotificationList. На этой странице мы будем отображать список оповещений, созданных приложением. Отложим на время дизайн этой страницы и вернёмся к основной странице приложения.

В зависимости от того, какой тип оповещения создаётся, поле ввода для заголовка оповещения и соответствующий текстовый блок с названием поля либо отображаются, либо нет. По умолчанию, выбрано предупреждение и эти элементы управления не отображаются. Нам необходимо обрабатывать изменения типа создаваемого оповещения и либо отображать, либо не отображать этот блок.

Если мы назначим обработчики изменения состояний радио-кнопок в XAML коде, может сложиться такая ситуация, что они сработают ещё до того, как создадутся элементы управления, которые мы хотим скрывать/показывать. Поэтому, мы добавим в код обработчик события Loaded страницы, а уже в него добавим регистрацию обработчиков изменения состояния радио-кнопок:

```
// Constructor
        public MainPage()
        {
            InitializeComponent();

            this.Loaded += new RoutedEventHandler(MainPage_Loaded);
        }

        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            AlarmType.Checked +=new RoutedEventHandler(AlarmType_Checked);
            ReminderType.Checked +=new RoutedEventHandler(ReminderType_Checked);
        }
```

И напишем обработчики событий радио-кнопок:

```
private void AlarmType_Checked(object sender, RoutedEventArgs e)
        {

            NotificationTitleCaption.Visibility = System.Windows.Visibility.Collapsed;
            NotificationTitle.Visibility = System.Windows.Visibility.Collapsed;

        }

        private void ReminderType_Checked(object sender, RoutedEventArgs e)
        {
            NotificationTitleCaption.Visibility = System.Windows.Visibility.Visible;
            NotificationTitle.Visibility = System.Windows.Visibility.Visible;
        }
```

Кнопка «посмотреть список» - позволяет перейти на страницу оповещений, зарегистрированных приложением. Добавим в XAML ссылку на обработчик события Click и напишем уже знакомый по предыдущим примерам код.

В результате XAML для кнопки будет выглядеть следующим образом:

```
<Button Content="посмотреть список" Name="CheckNotification" FontSize="22.667" Width="450" Cl
ick="CheckNotification_Click" />
```

А код, следующим образом:

```
1. private void CheckNotification_Click(object sender, RoutedEventArgs e)
2.         {
3.             NavigationService.Navigate(new Uri("/NotificationList.xaml",UriKind.RelativeOrAbs
   olute));
4.         }
```

Теперь осталось добавить код, который собственно и создаёт предупреждения и напоминания.

Для упрощения кода, разрешим нашему приложению в каждый момент времени иметь только одно предупреждение и одно напоминание. Зададим для них имена (должны быть уникальными среди всех, создаваемых приложением оповещений) в виде констант в классе:

```
1. const string MY_ALARM = "My Alarm";
2. const string MY_REMINDER = "My Reminder";
```

Обратите внимание, что для использования объектов Alarm и Reminder необходимо добавить следующую запись в блок using:

```
1. using Microsoft.Phone.Scheduler;
```

Теперь напишем две простые функции, которые будут создавать Alarm и Reminder соответственно.

```
1. private void CreateAlarm()
2.         {
3.             DateTime date = (DateTime)NotificationDate.Value;
4.             DateTime time = (DateTime)NotificationTime.Value;
5.             DateTime beginTime = date + time.TimeOfDay;
6.
7.             if (beginTime < DateTime.Now)
8.             {
9.                 MessageBox.Show("Указанное время оповещения уже прошло!");
10.                return;
11.            }
12.
13.            if (ScheduledActionService.Find(MY_ALARM) != null)
14.                ScheduledActionService.Remove(MY_ALARM);
15.
16.            Alarm myAlarm = new Alarm(MY_ALARM);
17.
18.            myAlarm.Content = NotificationText.Text;
19.
20.
21.            myAlarm.RecurrenceType = RecurrenceInterval.None;
22.            myAlarm.BeginTime = beginTime;
23.            myAlarm.ExpirationTime = beginTime.AddMinutes(5);
24.
25.            ScheduledActionService.Add(myAlarm);
26.
27.        }
28.
29.
30.        private void CreateReminder()
31.        {
32.            DateTime date = (DateTime)NotificationDate.Value;
33.            DateTime time = (DateTime)NotificationTime.Value;
34.            DateTime beginTime = date + time.TimeOfDay;
35.
36.            if (beginTime < DateTime.Now)
37.            {
38.                MessageBox.Show("Указанное время оповещения уже прошло!");
39.                return;
40.            }
41.
42.            if (ScheduledActionService.Find(MY_REMINDER) != null)
43.                ScheduledActionService.Remove(MY_REMINDER);
44.
45.            Reminder myReminder = new Reminder(MY_REMINDER);
46.            myReminder.Title = NotificationTitle.Text;
47.            myReminder.Content = NotificationText.Text;
48.
49.
50.            myReminder.RecurrenceType = RecurrenceInterval.None;
51.            myReminder.BeginTime = beginTime;
52.            myReminder.ExpirationTime = beginTime.AddMinutes(5);
53.            myReminder.NavigationUri = new Uri("/NotificationList.xaml",UriKind.RelativeOrAbs
```

```
    olute);


            ScheduledActionService.Add(myReminder);

        }
```

Давайте взглянем на код функций поближе.

Перед созданием мы выполняем простую проверку, что время, на которые мы хотим назначить оповещение еще не прошло:

```
DateTime date = (DateTime)NotificationDate.Value;
            DateTime time = (DateTime)NotificationTime.Value;
            DateTime beginTime = date + time.TimeOfDay;

            if (beginTime < DateTime.Now)
            {
                MessageBox.Show("Указанное время оповещения уже прошло!");
                return;
            }
```

Затем, поскольку по дизайну приложения, мы можем иметь только одно предупреждение и одно оповещение, мы ищем, присутствует ли уже ранее созданное оповещение и если оно присутствует, удаляем его.

Ниже, для примера приведён пример для Alarm:

```
if (ScheduledActionService.Find(MY_ALARM) != null)
                ScheduledActionService.Remove(MY_ALARM);
```

Дальше, в обоих функциях создаётся однократное оповещение, которое закончится через 5 минут, после указанного времени. При этом, для типа Reminder мы задаём заголовок и URI перехода, при нажатии на область уведомления.

```
Reminder myReminder = new Reminder(MY_REMINDER);
            myReminder.Title = NotificationTitle.Text;
            myReminder.Content = NotificationText.Text;


            myReminder.RecurrenceType = RecurrenceInterval.None;
            myReminder.BeginTime = beginTime;
            myReminder.ExpirationTime = beginTime.AddMinutes(5);
            myReminder.NavigationUri = new Uri("/NotificationList.xaml",UriKind.RelativeOrAbs
    olute);


            ScheduledActionService.Add(myReminder);
```

Теперь осталось добавить в XAML код ссылку на обработчик события Click:

```
<Button Content="установить" Name="SetNotification" FontSize="22.667" Width="450" Click="Set
    Notification_Click" />
```

И добавить в код простой обработчик:

```
private void SetNotification_Click(object sender, RoutedEventArgs e)
        {
            if ((bool)AlarmType.IsChecked)
                CreateAlarm();

            if ((bool)ReminderType.IsChecked)
                CreateReminder();
        }
```

Соберите приложение, запустите его (F5) и проверьте, как оно работает.

Сделаем теперь возможность просматривать созданные приложение оповещения.

Перейдем к странице NotificationList.xaml.

Используя наши знания по связыванию данных и шаблонам представления (см. SimpleRussianRSSReader), напишем XAML код отображения данных по оповещениям:

У меня в результате получился следующий XAML код (внутри тега phone:PhoneApplicationPage)

```
<!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="МЕНЕДЖЕР ОПОВЕЩЕНИЙ" Style="{StaticRes
ource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="список" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock Text="нет зарегистрированных оповещений" Name="NoNotifications" Visibi
lity="Collapsed" FontSize="22.667"/>
            <ListBox Name="Notifications">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <Grid Background="Transparent" Margin="0,0,0,30">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="380"/>
                                <ColumnDefinition Width="50"/>
                            </Grid.ColumnDefinitions>
                            <Grid Grid.Column="0">

                                <StackPanel Orientation="Vertical">
                                    <TextBlock Text="{Binding Title}" TextWrapping="NoWrap" F
oreground="{StaticResource PhoneAccentBrush}" FontWeight="Bold"/>
                                    <TextBlock Text="{Binding Content}" TextWrapping="Wrap" F
oreground="{StaticResource PhoneAccentBrush}"/>

                                    <StackPanel Orientation="Horizontal">
                                        <TextBlock Text="начало "/>
                                        <TextBlock Text="{Binding BeginTime}" HorizontalAlign
ment="Right"/>
                                    </StackPanel>
                                    <StackPanel Orientation="Horizontal">
                                        <TextBlock Text="окончание "/>
                                        <TextBlock Text="{Binding ExpirationTime}" Horizontal
Alignment="Right"/>
                                    </StackPanel>
                                    <StackPanel Orientation="Horizontal">
                                        <TextBlock Text="повторение "/>
                                        <TextBlock Text="{Binding RecurrenceType}" Horizontal
Alignment="Right"/>
                                    </StackPanel>
                                    <StackPanel Orientation="Horizontal">
                                        <TextBlock Text="в расписании? "/>
                                        <TextBlock Text="{Binding IsScheduled}" HorizontalAli
gnment="Right"/>
                                    </StackPanel>
                                </StackPanel>
                            </Grid>
                            <Grid Grid.Column="1">
                                <Button Tag="{Binding Name}" Content="X" Background="{Static
Resource PhoneBackgroundBrush}" Foreground="Red" VerticalAlignment="Top" BorderThickness="0"
 Width="50" Padding="0,0,0,0" Click="DeleteNotification_Click"/>
                            </Grid>
                        </Grid>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
    </Grid>
```

В связывании (Binding) используются поля объекта ScheduledNotofcation, базового для Alarm и Reminder.

Обратите внимание, что для использования объектов Alarm и Reminder необходимо добавить следующую запись в блок using:

using Microsoft.Phone.Scheduler;
Обратите внимание, что в XAML коде предусмотрена кнопка удаления конкретного оповещения. В поле Tag объекта Button мы пишем имя (Name) оповещения, которое уникально в рамках приложения. Это позволяет нам быстро написать обработчик этой кнопки:

```
private void DeleteNotification_Click(object sender, RoutedEventArgs e)
        {
            string NotificationName = (string)((Button)sender).Tag;

            ScheduledAction myAction = ScheduledActionService.Find(NotificationName);

            if (myAction != null)
            {
                ScheduledActionService.Remove(NotificationName);

            }
        }
```

Теперь нужно написать функцию, которая будет получать список всех оповещений приложения и устанавливать связывания данных для объекта ListBox Notifications.

Получить перечисляемый список объектов оповещения может следующая функция:

ScheduledActionService.GetActions<ScheduledNotification>();
То есть код получения списка и связывания с ListBox будет выглядеть следующим образом:

IEnumerable<ScheduledNotification> notifications =
ScheduledActionService.GetActions<ScheduledNotification>();
Notifications.ItemsSource = notifications;
Выделим его в отдельную функцию и добавим обработку состояния, когда у на не зарегистрировано ни одного оповещения:

```
private void InitNotofocationsList()
        {
            IEnumerable<ScheduledNotification> notifications = ScheduledActionService.GetActio
    ns<ScheduledNotification>();

            if (notifications.Count<ScheduledNotification>() > 0)
                NoNotifications.Visibility = System.Windows.Visibility.Collapsed;
            else
                NoNotifications.Visibility = System.Windows.Visibility.Visible;

            Notifications.ItemsSource = notifications;
        }
```

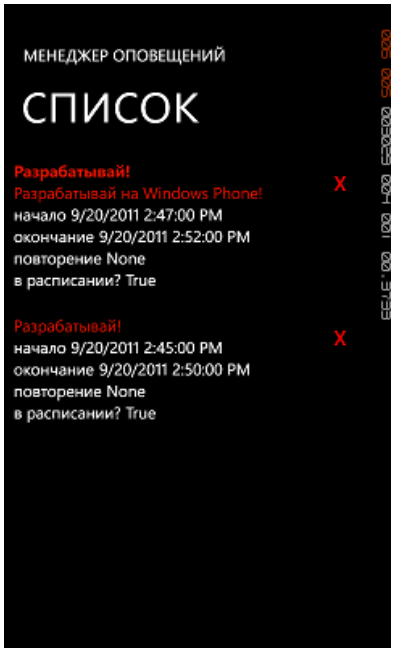
Поскольку, после удаления оповещения нам необходимо обновить список, добавим эту функцию в обработчик кнопки удаления оповещения:

```
private void DeleteNotification_Click(object sender, RoutedEventArgs e)
        {
            string NotificationName = (string)((Button)sender).Tag;

            ScheduledAction myAction = ScheduledActionService.Find(NotificationName);

            if (myAction != null)
            {
                ScheduledActionService.Remove(NotificationName);

                InitNotofocationsList();
            }
        }
```

Дальше, переопределим метод OnNavigatedTo и добавим в него вызов функции InitNotificationList():

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);

            InitNotofocationsList();
        }
```

Соберите приложение, запустите его (F5) и проверьте работоспособность.



В качестве самостоятельного упражнения, модифицируйте код приложения таким образом, чтобы оно могло создавать больше чем 2 оповещения. Добавьте более тщательную обработку ошибок и ввода пользователя.

Живые тайлы (Live Tiles)

Для начала определимся с тем, что такое тайл (Tile). Тайл – это ссылка на приложение, которая отображается на панели старт. Бывает два вида тайлов. Первый - это тайл приложения (Application Tile), который появляется, когда пользователь закрепляет (pin) приложение в панели старт. Он один и есть у приложения всегда, даже когда ссылка на приложение не отображается в стартовой панели. Второй вид – это вторичные тайлы (Secondary Tiles), которые создаются програмно приложением по запросу пользователя. Например, если это новостное приложение, оно может предлагать возможность создать вторичный тайл, который будет относиться к какому-нибудь отдельному новостному разделу, например, мобильная разработка, веб-разработка и т.д. При этом, обычно, нажатие на такой тайл приводит пользователя не на основную страницу приложения, а на некую выделенную. При этом такой тайл может обновляться отличным от тайла приложения образом: например, показывать количество новых новостей не всего, а по определённой тематике.

Удобно представить информацию о различии в типах тайлов в виде таблицы:

	Application Tile	Secondary Tiles
Как создаётся	Только пользователем, который прикрепляет приложение (pin to start) из списка в стартовую панель. Начальные свойства тайла приложения могут быть заданы в манифесте приложения.	Пользователем в приложении, когда он выбирает создать дополнительный тайл (Create(Uri, ShellTileData) API)
Как удаляется	Не удаляется. Вне зависимости от присутствия на стартовой панели. Может обновляться также вне зависимости от присутствия на стартовой панели.	Пользователем, если он открепляет тайл от стартовой панели. При де-инсталляции приложения. Используя Delete() API
Как обновляется	ShellTile API ShellTileSchedule API Push Notifications	ShellTile API ShellTileSchedule API Push Notifications

Обратите внимание, что Application Tile всегда первый в коллекции ActiveTiles, вне зависимости от того, закреплён ли он в стартовой панели.

Тайлы в Windows Phone – двухсторонние. Если у тайла определены обе стороны, и фронтальная и обратная, при отображении, между ними происходит переключение. Если обратная сторона не определена (не установлено не одно из свойств обратной стороны), переключения между ними не

происходит.

Ниже, показан пример фронтальной стороны тайла

В правом верхнем углу тайла, в круге, отображается свойство Count, числовое значение от 1 до 99. Если свойство Count не определёно или установлено в 0, не отображается.

В нижней части тайла находится заголовок – Title. Это строка, она должна помещаться в размер тайла, то есть максимальная длина около 15 символов.

Сам тайл заполнен фоновым рисунком – BackgroundImage.

Обратная сторона тайла выглядит похожим образом:

В нижней части тайла находится заголовок обратной стороны тайла – BackTitle. Это строка, она должна помещаться в размер тайла, то есть максимальная длина около 15 символов.

В верхней части стороны находится содержание обратной стороны тайла – BackContent. Это строка, помещается около 40 символов.

Сам тайл заполнен фоновым рисунком – BackBackgroundImage.

Обратите внимание, что при создании вторичного тайла (Secondary Tile) фоновые рисунки для фронтальной и обратной стороны тайла должны быть в локальных ресурсах. При обновлении могут использоваться и локальные и удалённые ресурсы.

Перейдём к практическому знакомству с тайлами. Создадим новое приложение на базе стандартного шаблона Windows Phone Application и назовём приложение LiveTilesExample.

В этом приложении мы научимся изменять тайл приложения, создавать вторичный тайл и обновлять фоновый рисунок фронтальной стороны тайла по расписанию с использованием ShellTileSchedule API.

Добавьте в приложение 2 картинки формата PNG и размера 173 X 173. Одну назовите front.png, а вторую back.png. После добавления картинок, установите для них тип сборки (Build Action) в Content (по умолчанию Resource).

После этого, добавьте ещё одну страницу в приложение и назовите её SecondaryPage.xaml. После того, как её добавите измените XAML код в ней следующим образом:

```
<!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="LIVE TILES" Style="{StaticResource Ph
oneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="другая" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>
```

Перейдите к MainPage.xaml. Итак, мы попробуем обновить свойства тайла приложения, создать вторычный тайл и обновить фоновую картинку фронтальной стороны тайла приложения. Добавим три кнопки с соответствующими названиями в область контента страницы:

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <Button Name="AppTile" Content="изменить тайл приложения" />
                <Button Name="SecTile" Content="установить вторичный тайл" />
                <Button Name="UpdateAppTile" Content="обновление тайла приложения" />
```

```
            </StackPanel>
        </Grid>
```

В окне дизайнера щелкнем двойным щелчком по кнопке «изменить тайл приложения», будет автоматически создан обработчик события Click и мы перейдём в редактор кода.

Для работы с тайлами, в блок usingнеобходимо добавить следующую запись:

using Microsoft.Phone.Shell;

Перейдём в код метода обработки события Click кнопки AppTile и добавим код, который изменяет вид тайла приложения:

```
private void AppTile_Click(object sender, RoutedEventArgs e)
        {
            ShellTile apptile = ShellTile.ActiveTiles.First();

            StandardTileData appTileData = new StandardTileData();
            appTileData.Title = "Тайл приложения";
            appTileData.Count = 5;
            appTileData.BackgroundImage = new Uri("/front.png", UriKind.RelativeOrAbsolute);

            appTileData.BackTitle = "Обратная сторона";
            appTileData.BackContent = "Очень важное сообщение";
            appTileData.BackBackgroundImage = new Uri("/back.png", UriKind.RelativeOrAbsolute
    );

            apptile.Update(appTileData);

        }
```

Обратите внимание, что тайл приложения всегда первый в активном списке тайлов, даже если он не закреплён в стартовой панели.

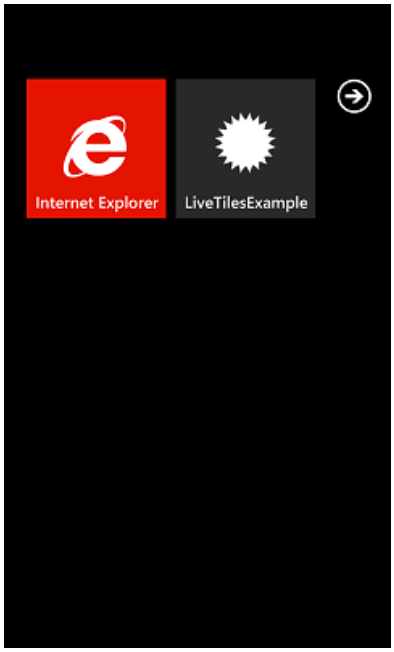
Соберите приложение и разверните его (Deploy Solution).

В эмуляторе, перейдите в список устройств и закрепите приложение на стартовой панели (долгое нажатие на приложение в списке и далее pin to start).

pin to start

uninstall

Обратите внимание, как выглядит тайл приложения на стартовой панели:



Запустите приложение из стартовой панели и нажмите на кнопку кнопке «изменить тайл приложения». Далее нажмите на кнопку Back на телефоне. Обратите внимание, что тайл приложение изменился и имеет 2 стороны, которые меняются со временем.

Закройте эмулятор, чтобы в следующий раз запускать приложение на чистом устройстве.

В окне дизайнера щелкнем двойным щелчком по кнопке «установить вторичный тайл», будет автоматически создан обработчик события Click и мы перейдём в редактор кода.

Создадим вторичный тайл, по нажатию на который, пользователь будет сразу попадать на вторую страницу приложения:

```
 1. private void SecTile_Click(object sender, RoutedEventArgs e)
 2.         {
 3.             StandardTileData secTileData = new StandardTileData();
 4.             secTileData.Title = "Вторичный тайл";
 5.             secTileData.Count = 5;
 6.             secTileData.BackgroundImage = new Uri("/back.png", UriKind.RelativeOrAbsolute);
 7. 
 8.             secTileData.BackTitle = "Обратная сторона";
 9.             secTileData.BackContent = "Просто сообщение";
10.             secTileData.BackBackgroundImage = new Uri("/front.png", UriKind.RelativeOrAbsolut
    e);
11. 
12.             ShellTile.Create(new Uri("/SecondaryPage.xaml", UriKind.RelativeOrAbsolute), secT
```

```
    ileData);
13.         }
```

Запустите приложение (F5).

В приложении, нажмите кнопку «установить вторичный тайл». Обратите внимание, что приложение автоматически перешло в стартовую панель, куда уже добавлен вторичный тайл. Нажмите на него и убедитесь, что отображается вторая страница приложения.

Перейдем обратно в Visual Studio и в дизайнере основной страницы щелкнем двойным щелчком по кнопке «обновление тайла приложения», будет автоматически создан обработчик события Click и мы перейдём в редактор кода.

Добавим код в обработчик. Мы обновляем один раз, через две минуты после нажатия кнопки стартуем.

```
 1. private void UpdateAppTile_Click(object sender, RoutedEventArgs e)
 2.         {
 3.             ShellTileSchedule appTileSchedule = new ShellTileSchedule();
 4.
 5.             appTileSchedule.Recurrence = UpdateRecurrence.Onetime;
 6.
 7.
 8.             appTileSchedule.StartTime = DateTime.Now.AddMinutes(2);
 9.
10.             appTileSchedule.RemoteImageUri = new Uri("http://
    замените на путь к вашей картинке на удалённом сервере");
11.             appTileSchedule.Start();
12.
13.         }
```

Замените в коде путь к картинке на уделённом сервере. Обратите внимание, что поддерживается только HTTP, размер картинки должен быть не более 80K и скачиваться она должна не более 30 секунд. Поскольку задача скачивания ставится в пакет, то картинка может обновиться в течение часа.

Эта картинка заменит фоновый рисунок фронтальной стороны тайла приложения.

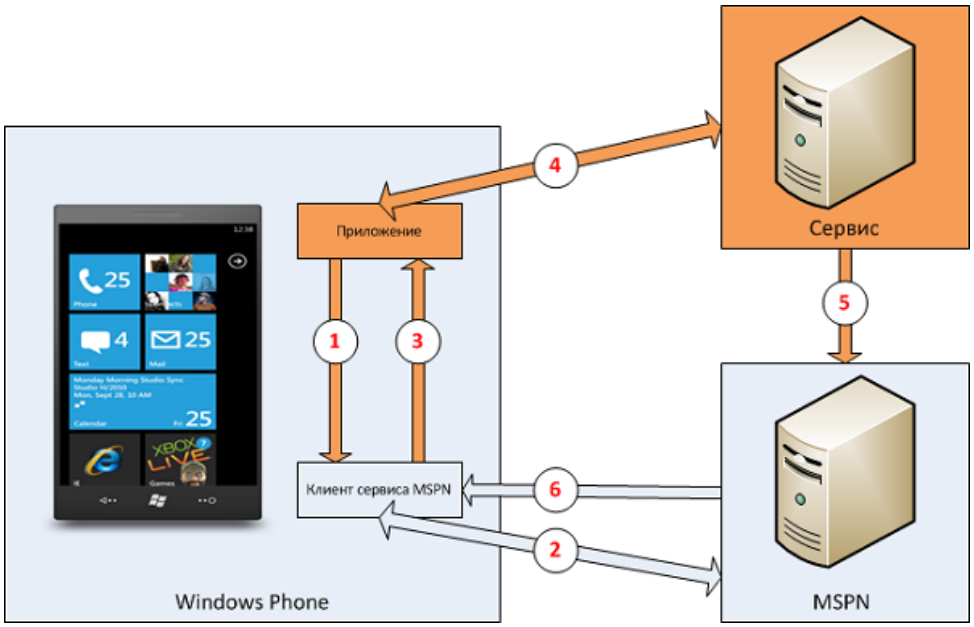
Самостоятельно добавьте кнопку и обработчик, чтобы заменить фоновый рисунок фронтальной стороны вторичного тайла (будет работать только когда тайл закреплён в стартовой панели, используйте конструктор ShellTileSchedule с параметром).

Также попробуйте создать к приложению фоновый агент (Background Agent), который будет обновлять тайлы, используя ShellTile API.

Push Notification

Возможность Push оповещений, т.е. получения информации об изменении какого-то внешнего ресурса, без необходимости его постоянно опрашивать (Poll). Этот механизм реализован через специальный внешний сервис – Microsoft Push Notification Service, который и отвечает за отсылку Push оповещения на устройство.

Схематично работа с сервисом Microsoft Push Notification Service представлена на следующем рисунке.



Приложение, которое хочет использовать Push оповещения, запрашивает URI у клиента Push сервиса оповещений (1), клиент Push сервиса взаимодействует с Microsoft Push Notification Service (MSPN), получает от него URI(2) и отдаёт его приложению (3). Приложение передаёт этот URI своему сервису, который будет формировать сообщения приложению и отсылать их методом POST в сервис MSPN (5) по указанному URI. Когда сервис MSPN получает сообщение, он запускает процедуру доставки его на устройство (6).

Обратите внимание, что сервис MSPN не оповещает сервис, формирующий сообщения о том, когда сообщение получено устройством.

Прежде чем перейти к типам оповещений, перечислим несколько базовых технических ограничений:

- на одно приложение – один канал Push оповещений;
- не более 30 каналов Pushоповещений на устройство;
- максимальный размер оповещения: 1 Кб для заголовка и 3 Кб на содержимое.

Всего доступно три типа Push оповещений:

1. Toast;
2. Tile;
3. Raw.

Приложение может зарегистрировать на получение Toast оповещение. Тогда, если приложение запущено, и приходит Toast оповещение, оно не отображается в верхней части экрана. Если приложение не зарегистрировано на получение Toast оповещения или не запущено, то Toast оповещение, отображается в верхней части экрана, сообщая пользователю о том, что что-то требует его внимания. При нажатии на оповещение запускается приложение. Разработчику доступны Deep Toast оповещения, когда при запуске открывается не стартовая страница приложения, а определённая и с определёнными параметрами. Пример Toast оповещения – оповещение о приходе SMS.

Основное назначение Tile оповещения – обновление тайлов приложения, закреплённых на стартовой панели. Приложение не получает Tile оповещение.

Оповещение типа Raw, позволяет отправить сообщение работающему приложению. Если приложение на момент получения не запущено, сообщение не будет отправлено на устройство.

Перейдём к практическому знакомству с возможностями Push Notification. Создадим новое приложение на базе стандартного шаблона Windows Phone Application и назовём приложение PushNotificationExample.

Как понятно, необходим какой-то сервис, чтобы посылать сообщения по полученному URI в MPNS. Мы не будем останавливаться здесь на этом подробно, а просто воспользуемся примерами тестовых ASP.NET приложений:

- Toast: http://msdn.microsoft.com/en-us/library/hh202967(v=VS.92).aspx
- Tile: http://msdn.microsoft.com/en-us/library/hh202970(v=VS.92).aspx
- Raw: http://msdn.microsoft.com/en-us/library/hh202977(v=VS.92).aspx

В примере к этому разделу все три страницы включены в один проект и объединены в одно решение с проектом PushNotificationExample для Windows Phone.

Вернёмся к приложению для Windows Phone. У него будет очень простой функционал. Приложение будет регистрировать канал и на нём ожидать всех типов оповещений: Toast, Tile и Raw.

Сначала слегка изменим и дополним дизайн MainPage.xaml:

```
<!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="PUSH NOTIFICATION" Style="{StaticReso
urce PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="настройки" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <Button Name="SetPushNotification" Content="установка оповещений" />

            </StackPanel>
        </Grid>
```

Двойным щелчком по кнопке «установка оповещений» в коде дизайнера создадим и перейдём в

обработчик события Click кнопки.

Добавим код, который ищет канал с имеем «MyChannel», если не находит, то создаёт, после чего регистрируется на обработку событий Toast и Raw (Tile сообщение до приложения не доходит, оно напрямую меняет тайл).

```
private void SetPushNotification_Click(object sender, RoutedEventArgs e)
        {
            HttpNotificationChannel myChannel;
            bool isNewChannel = false;
            string myChannelName = "MyChannel";

            myChannel = HttpNotificationChannel.Find(myChannelName);

            if (myChannel == null)
            {
                myChannel = new HttpNotificationChannel(myChannelName);
                isNewChannel = true;

            }

            //получаем URI и его изменения
            myChannel.ChannelUriUpdated += new EventHandler<NotificationChannelUriEventArgs>(myToastChannel_ChannelUriUpdated);

            //обрабатываем ошибки
            myChannel.ErrorOccurred += new EventHandler<NotificationChannelErrorEventArgs>(myToastChannel_ErrorOccurred);

            //регируем на получение Toast оповещения при запущеном приложении
            myChannel.ShellToastNotificationReceived += new EventHandler<NotificationEventArgs>(myToastChannel_ShellToastNotificationReceived);

            //регируем на получение Raw оповещения при запущеном приложении
            myChannel.HttpNotificationReceived += new EventHandler<HttpNotificationEventArgs>(myToastChannel_HttpNotificationReceived);

            if (isNewChannel)
                myChannel.Open();

            // говорим о том, что у нас будут toast оповещения
            if (!myChannel.IsShellToastBound)
                myChannel.BindToShellToast();

            // говорим о том, что у нас будут tile оповещения
            if (!myChannel.IsShellTileBound)
                myChannel.BindToShellTile();

        // если канал был раньше - посылаем URI сервису
        if (!isNewChannel)
                SendURIToService(myChannel.ChannelUri);

        }
```

Добавим обработчики событий ошбок и получения/изменения URI

```
void myToastChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs e)
        {
            // здесь мы должны были бы обработать ошибку
            // здесь просто выводим, для использования в тестировании
            Dispatcher.BeginInvoke(() =>
            {
                System.Diagnostics.Debug.WriteLine(e.Message);
                MessageBox.Show(String.Format("Error: {0}", e.Message));
            });
        }

        void myToastChannel_ChannelUriUpdated(object sender, NotificationChannelUriEventArgs e)
        {
            // здесь мы должны были бы отослать URI нашему сервису
            // просто выводим, для использования в тестировании
            // из-за взаимодействия с UI - используем Dispatcher
            Dispatcher.BeginInvoke(() => SendURIToService(e.ChannelUri));
        }
```

В нашем случае, функция SendURIToService() просто отображает сообщение в UI, поэтому и необходимо

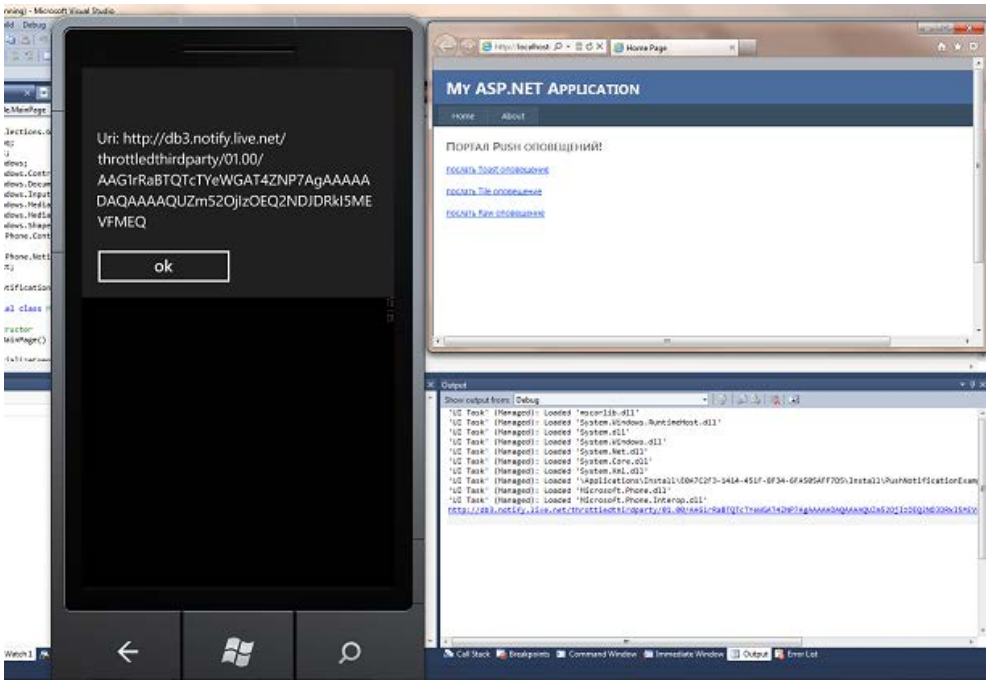
обернуть её в диспатчер:

```
void SendURIToService(Uri channelURI)
        {
            // здесь мы должны были бы отослать URI нашему сервису
            // просто выводим, для использования в тестировании
            System.Diagnostics.Debug.WriteLine(channelURI.ToString());
            MessageBox.Show(String.Format("Uri: {0}", channelURI.ToString()));
        }
```

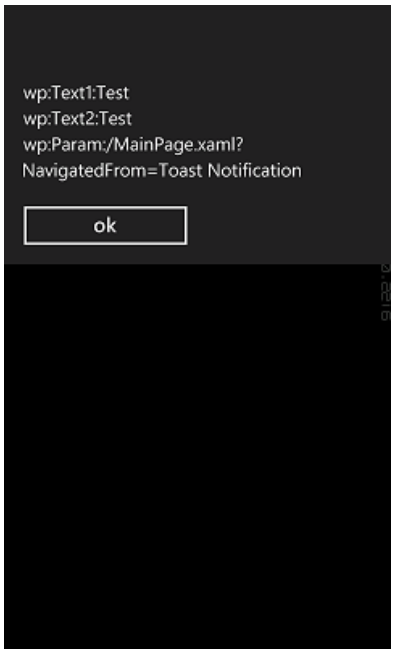
В завершении, добавим обработчик Toast и Raw оповещений, который будут работать при запущенном приложении:

```
void myToastChannel_ShellToastNotificationReceived(object sender, NotificationEventArgs e)
        {
            // здесь мы должны были бы обработать сообщение
            // здесь просто выводим, для использования в тестировании

            StringBuilder sb = new StringBuilder();

            foreach (string key in e.Collection.Keys)
            {
                sb.AppendFormat("{0}:{1}\n", key, e.Collection[key]);
            }

            string result = sb.ToString();

            Dispatcher.BeginInvoke(() =>
            {
                System.Diagnostics.Debug.WriteLine(result);
                MessageBox.Show(result);
            });
        }

void myToastChannel_HttpNotificationReceived(object sender, HttpNotificationEventArgs e)
        {
            //добавьте свой обработчик Raw сообщения, который будет его разбирать
            Dispatcher.BeginInvoke(() =>
            {
                System.Diagnostics.Debug.WriteLine("Получено Raw сообщение.");
                MessageBox.Show("Получено Raw сообщение");
            });

        }
```

Запустите решение (F5), откройте страницу ASP.NET проекта в браузере. В окне приложения, нажмите на кнопку «установка оповещений» и дождитесь отображения URI в окне вывода отладочной информации:



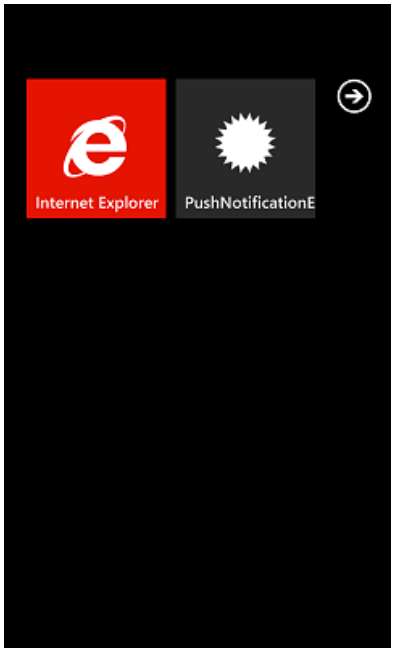
Закройте диалоговое окно приложения. Скопируйте URI из окна вывода отладочной информации и перейдите на страницу Toast оповещения, введите скопированный URI, введите Title и Subtitle и, нажав на кнопку, отошлите сообщение. Поскольку приложение запущено, оно самостоятельно обработает сообщение и выведет диалоговое окно:



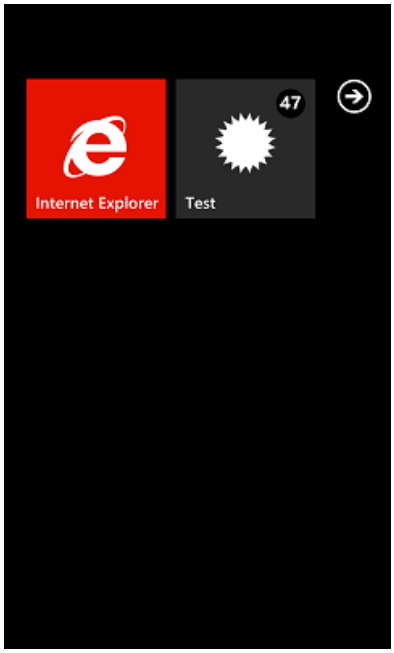
Toast сообщение не отобразится.

Закройте приложение, нажав кнопку Back (отладка завершится), перейдите к веб-приложению и ещё раз отправьте Toast сообщение. Теперь оно отобразится в интерфейсе пользователя.

Перейдите к списку приложений и закрепите приложение PushNotificationExample в стартовую панель (pin to start).



Перейдите к веб-приложению на страницу отправки Tile сообщения и отправьте Tile сообщение, указав Front Title и Count (не забудьте указать URI). Обратите внимание, как изменился тайл приложения.



В качестве самостоятельно работы, запустите приложение и проверьте, что оно получает Raw сообщения.

Добавьте в приложения картинки и сформируйте Tile сообщение, чтобы они использовались в обновлённом тайле.

Итоги и следующие шаги

В этой статье мы познакомились с возможностями системы, которые позволяют нам оповещать пользователя, напоминать ему или даже предоставлять саму востребованную информацию без необходимости запускать приложения.

В следующих статьях мы познакомимся с новым API работы с камерой, и работой с дополнительными аппаратными возможностями устройства, такими как компас, гироскоп и Motion-сенсор.

Файлы для загрузки

Проект SimpleNotificationManager

Проект LiveTilesExample

Проект PushNotificationExample

Всё о Splash Screen в Windows Phone

Рейтинг статьи

Какое бы вы приложение не писали, даже самое простое, его запуск на телефоне будет занимать некоторое время, поэтому хорошей идеей будет показывать какой-нибудь Splash Screen во время загрузки. Существует несколько вариантов использования Splash Screen:

- Использовать изображение
- Использовать анимированный Splash Screen
- Не использовать Splash Screen

Давайте поподробнее остановимся на каждом из этих пунктов.

Использование изображения

При создании проекта в Visual Studio у вашего приложения уже есть Splash Screen по-умолчанию:

Для замены его на свой Splash Screen необходимо просто добавить в проект файл с именем SplashScreenImage.jpg (**внимание**: имя должно быть именно таким). Также у вашего Splash Screen разрешение должно быть 480x800px (480 пикселей в ширину и 800, соответственно, в высоту), и не забудьте указать в свойствах изображения Build Action: Content.

Использование анимированного Splash Screen

Если загрузка первой страницы вашего приложения занимает некоторое время (например, используются какие-то данные из Интернета), то создание анимированного экрана загрузки – так же хорошая идея. Итак, алгоритм работы будет следующий:

1. При загрузке приложения показывается SplashScreenImage.jpg
2. После загрузки открывается **MainPage** с открытым **Popup** окном, развернутым на весь экран. Этот **Popup** может быть основан на вашем **SplashScreenImage** для создания наилучшего эффекта
3. Как только загрузка данных закончится – **Popup** закроется, и пользователь увидит содержимое **MainPage**.

Как вы можете заметить, термин «анимированный Splash Screen» на самом деле не совсем верен: мы используем обычный статический экран заставки, а потом просто показываем пользователю страницу, похожую на экран заставки, тем временем выполняя в фоне какие-нибудь действия.

Для того чтобы наша загрузка выполнялась в отдельном потоке мы будем использовать класс **BackgroundWorker**. Этот класс позволяет делать какую-то работу в фоне, оставляя UI отзывчивым для пользователя. Также у нас будет возможность отслеживать состояние фонового процесса и убрать **Popup** при наступлении какого-нибудь события.

Для начала давайте создадим свой UserControl, для примера назовем его **AnimatedSplashScreen**. Далее нам нужно добавить обработку **Popup** в MainPage.xaml.cs. Для этого добавим следующие пространства имен:

Дополнительные материалы

- Создание приложений для Windows Phone с использованием Silverlight+XNA
- Интеграция приложения на Windows Phone со SkyDrive
- Использование HTML5 и JavaScript для разработки приложений под Windows Phone
- Переключение слайдов презентации с использованием Windows Phone: UDP соединение
- Переключение слайдов презентации с использованием Windows Phone: TCP соединение
- Дизайн приложений для WP7. Темы и акценты
- Прокрустовы окна. Как вписаться в устройства с минимальными потерями (видео)
- HTML5 для разработки мобильных приложений (видео)
- appmakr: Как сделать приложение для Windows Phone за 10 минут и без программирования
- Разработка игр для Windows Phone (видео)
- Пример простого онлайн-проигрывателя
- Пару слов о Motion API
- Монетизация приложений — где деньги лежат, или почему это все реклама?
- Работа с HTTP в 3 строчки, или делаем свой RSS Reader с WebClient и WebBrowser
- Быстрое создание контент-оринетированых приложений
- Карты и гелокационные данные на Windows Phone
- Обновления Live Tiles в фоновых агентах
- Всё о Splash Screen в Windows Phone

```
using System.Threading;

using System.Windows.Controls.Primitives;

using System.ComponentModel;
```

Теперь необходимо создать BackgroundWorker и наш Popup:

```
BackgroundWorker backroungWorker;

        Popup myPopup;         // Constructor

        public MainPage()

        {

            InitializeComponent();

            myPopup = new Popup() { IsOpen = true, Child = new AnimatedSplashScreen() };

            backroungWorker = new BackgroundWorker();

            RunBackgroundWorker();

        }

           private void RunBackgroundWorker()

        {

            backroungWorker.DoWork += ((s, args) =>

            {

                Thread.Sleep(5000);

            });

               backroungWorker.RunWorkerCompleted += ((s, args) =>

            {

                this.Dispatcher.BeginInvoke(() =>

                {

                    this.myPopup.IsOpen = false;

                }

            );

            });

            backroungWorker.RunWorkerAsync();

        }
```

Что мы здесь делаем:

1. Создаем **Popup**, по-умолчанию открытый, который содержит наш UserContol – **AnimatedSplashScreen**;
2. Создаем **BackgroundWorker**, который в фоне замораживает поток на 5000 мс.
3. Как только 5 секунд проходит, **BackgroundWorker**переходит в состояние **RunWorkerCompleted** и наш **Popup** закрывается.

Основная логика есть, теперь давайте вернемся к содержимому нашего **Popup**. У себя в примере я использую изображение **SplashScreenImage**, **TextBlock** и **PerformanceProgressBar** из **WP7 Toolkit**.

```
<GRID x:name="LayoutRoot" width="480" height="800" background="White">

      <IMG stretch="Fill" source="/SplashScreenImage.jpg">
```

```
4.
5.      <TEXTBLOCK opacity="0" x:name="textBlock" height="66" verticalalignment="Bottom" textwr
    apping="Wrap" text="LOADING" margin="139,0,142,203" foreground="#FF7DA8B4" fontsize="48" fon
    tfamily="/SplashScreenTest;component/Fonts/Fonts.zip#Curlz MT">
6.
7.              <TOOLKIT:PERFORMANCEPROGRESSBAR width="456" height="11" background="#FF79A4B7"
     verticalalignment="Top" margin="12,603,0,0" foreground="#FF79A4B7" name="performanceProgres
    sBar1" isindeterminate="True" horizontalalignment="Left">
8.
9.              </TOOLKIT:PERFORMANCEPROGRESSBAR>
10.
11.     </TEXTBLOCK>
12.
13. </GRID>
```

Для того чтобы немного оживить загрузку, я в **Expression Blend** создал анимацию мигания для текста:

```
1.  <USERCONTROL.RESOURCES>
2.
3.      <STORYBOARD x:key="Blinking" repeatbehavior="Forever">
4.
5.          <DOUBLEANIMATIONUSINGKEYFRAMES storyboard.targetproperty="(UIElement.Opacity)" story
    board.targetname="textBlock">
6.
7.              <EASINGDOUBLEKEYFRAME value="0" keytime="0">
8.
9.              <EASINGDOUBLEKEYFRAME value="1" keytime="0:0:0.5">
10.
11.                 <EASINGDOUBLEKEYFRAME.EASINGFUNCTION>
12.
13.                     <CUBICEASE easingmode="EaseOut">
14.
15.                 </CUBICEASE></EASINGDOUBLEKEYFRAME.EASINGFUNCTION>
16.
17.             </EASINGDOUBLEKEYFRAME>
18.
19.             <EASINGDOUBLEKEYFRAME value="0" keytime="0:0:1">
20.
21.                 <EASINGDOUBLEKEYFRAME.EASINGFUNCTION>
22.
23.                     <CUBICEASE easingmode="EaseIn">
24.
25.                 </CUBICEASE></EASINGDOUBLEKEYFRAME.EASINGFUNCTION>
26.
27.             </EASINGDOUBLEKEYFRAME>
28.
29.         </EASINGDOUBLEKEYFRAME>
30.
31. </DOUBLEANIMATIONUSINGKEYFRAMES>
32.
33.     </STORYBOARD>
34.
35.  </USERCONTROL.RESOURCES>
```

Теперь осталось только добавить и запустить эту анимацию в конструкторе класса AnimatedSplashScreen:

```
1.  public AnimatedSplashScreen()
2.
3.   {
4.
5.      InitializeComponent();
6.
7.      Storyboard Blinking = this.Resources["Blinking"] as Storyboard;
8.
9.      Blinking.Begin();
10.
11.  }
```

Всё, наш анимированный **Splash Screen** готов! Запустив проект, мы увидим следующую картину:

Пример моего анимированного **Splash Screen** вы можете скачать по ссылке: SkyDrive.
В качестве альтернативы **Popup** можно использовать отдельную страницу, но тогда нужно предусмотреть чтобы при нажатии на кнопку **Назад** пользователь выходил из приложения, а не попадал снова на страницу загрузки. Это можно сделать, например, удалив страницу из стека навигации:

```
1. if (NavigationService.CanGoBack == true)
2.
3.   {
4.
5.   NavigationService.RemoveBackEntry();
6.
7.   }
```

Не использовать Splash Screen

Для того чтобы убрать у вашего приложения экран заставки, достаточно просто удалить файл SplashScreenImage.jpg из вашего решения.

Итак, мы рассмотрели различные варианты создания Splash Screen в Windows Phone. Надеюсь, что эта статья оказалась для вас полезной :).

3D Графика для Windows Phone 7 с использованием XNA

Рейтинг

Введение

На сегодняшний день большой популярностью пользуются приложения, использующие трехмерную графику. Наиболее очевидным примером таких приложений являются игры. Мы уже давно привыкли к тому, что игры для персональных компьютеров потрясают нас реалистичной трехмерной графикой и красотой визуальных эффектов, в то время как игры для мобильных телефонов пока значительно уступают своим старшим собратьям в визуальном плане.

С развитием мобильных технологий появилась возможность помещать мощные комплектующие в миниатюрные корпуса телефонов, из-за чего современные мобильные телефоны могут неплохо справляться с задачей рендеринга достаточно сложных трехмерных сцен.

В рамках данной статьи мы рассмотрим технологию Microsoft XNA Framework, которая позволяет разрабатывать приложения, поддерживающие трехмерную графику, для новых телефонов, основанных на Windows Phone 7.

Начало работы

Перед тем как переходить к основной части и начать рассматривать примеры стоит убедиться в том, что ваш компьютер удовлетворяет всем требованиям и установить необходимое программное обеспечение.

Как говорилось ранее, для работы с трехмерной графикой для Windows Phone 7 мы будем использовать XNA Framework 4. Разработка будет вестись в привычной для .NET разработчиков среде Visual Studio 2010 с дополнительной надстройкой Windows Phone Developer Tools для XNA Framework.

Системные требования Windows Phone Developer Tools для XNA Framework подробно описаны в Release Notes, здесь же будут описаны основные положения:

Операционная система:

Windows Vista SP2 (кроме редакции Starter), Windows 7 (кроме редакции Starter)

Примечание:

Разрабатывать приложения для Windows и Xbox 360 можно с использованием XNA Framework 4 и в Windows XP, однако разрабатывать приложения для Windows Phone 7 можно только в Windows Vista/7.

Железо:

Для работы эмулятора Windows Phone требуется видео-карта с поддержкой DirectX 10.1.

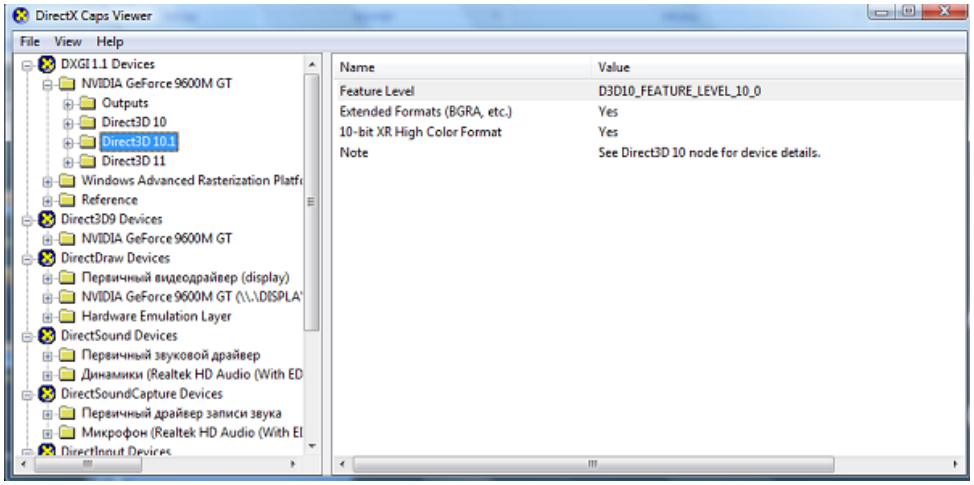
Примечание:

Информацию о своей видеокарте вы можете получить на сайте производителя.

Также рекомендуется обновить драйверы для видеокарты.

Если видеокарта поддерживает DirectX 10.1 и все настроено правильно, то зайдя в DirectX Caps Viewer (входит в состав DirectX SDK) в разделе DXGI 1.1 Devices\[Название вашей видеокарты] вы найдете подраздел Direct3D10.1.

Разработка 3D игр

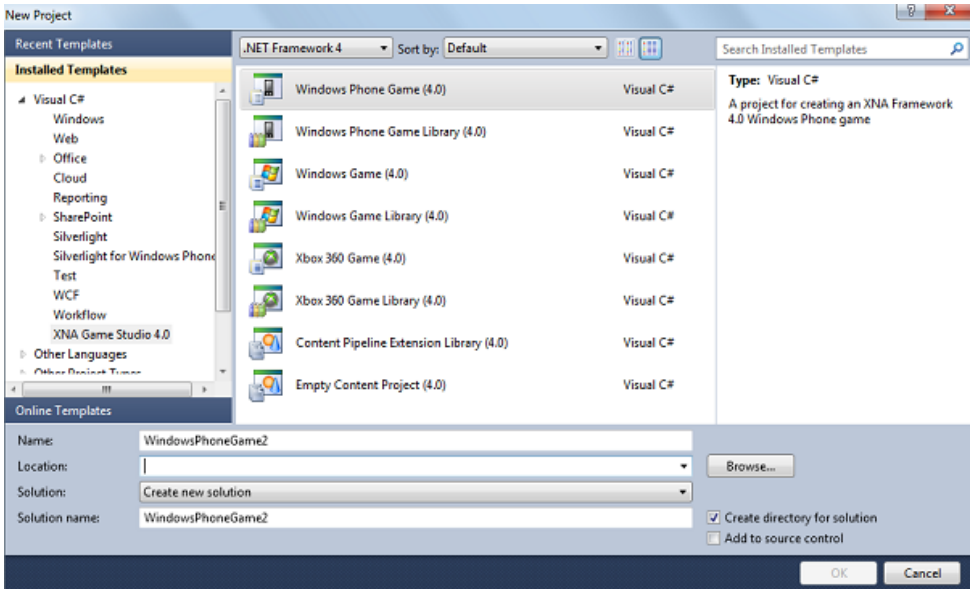


Если компьютер удовлетворяет всем описанным требованиям, то можно переходить к этапу установки инструментов для разработки.

Установите последнюю версию DirectX SDK.

УстановитеWindows Phone Developer Tools. (http://create.msdn.com/en-us/resources/downloads)

Теперь в Visual Studio создайте новый проект Windows Phone Game (4.0) из раздела XNA Game Studio 4.0.



Попробуйте запустить проект и, если все сделано правильно, то вы должны увидеть следующую картину (первый запуск приложения обычно занимает достаточно много времени).

Примечание:

Для того, чтобы сократить время следующего запуска приложения нужно не закрывать эмулятор, а просто остановить отладку приложения в Visual Studio.

Приступаем к работе. Основы XNA Framework

XNA Framework позволяет разрабатывать приложения для платформ Windows, Xbox 360 и Windows Phone 7. Причем эта технология специально создана таким образом, чтобы сделать разработку приложений под различные платформы максимально простой и удобной.

Следующая схема описывает принцип работы XNA Framework:

XNA Framework для Windows базируется на .NET Framework[] и на .NET Compact Framework[] для Xbox 360 и Windows Phone 7. Разработка ведется в Visual Studio 2010, для этих целей существует надстройка XNA Game Studio, которая добавляет в Visual Studio новые элементы, такие как шаблоны проектов XNA Framework.

На следующем рисунке показаны основные элементы XNA Framework.

XNA Framework содержит в себе элементы, которые необходимы для разработки игр, например, библиотека функций для работы с матрицами и векторами, библиотека для унифицированной работы с устройствами ввода и т.д. Более того, XNA Framework содержит дополнительные элементы (Extended Framework), которые решают многие вопросы, возникающие в процессе разработки игры.

Extended Framework состоит из Application Model и Content Pipeline.

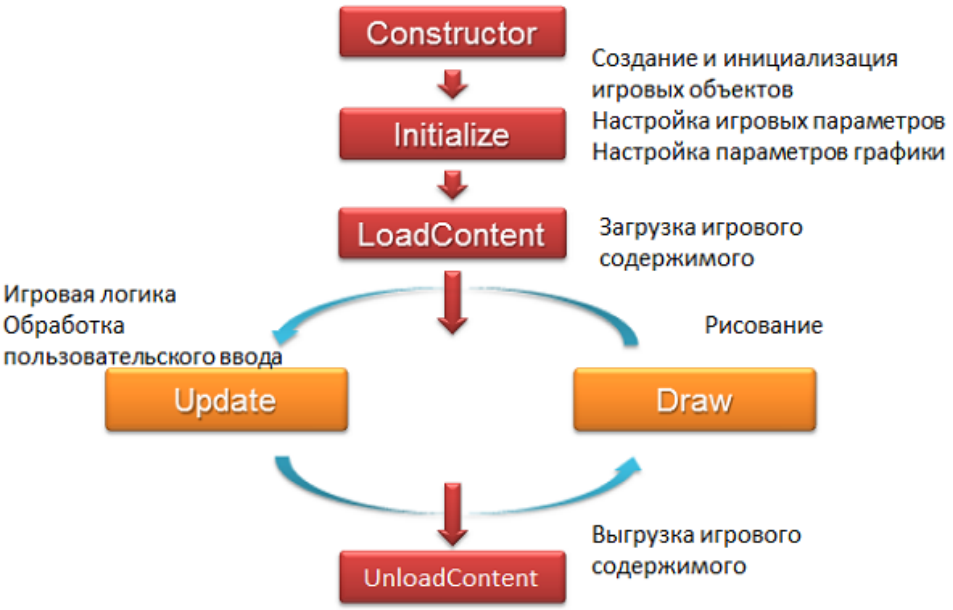
Application Model – каркас приложения. Каждый новый проект XNA Game уже содержит в себе класс Game1, которые имеет набор методов, каждый из которых имеет свое предназначение. Разработчику не нужно задумываться над такими вопросами как:

Как создать игровой цикл?

Когда обрабатывать пользовательский ввод?

Когда формировать изображение на экране?

Следующая схема описывается последовательность вызовов методов каркаса приложения:



Content Pipeline – средство, которое унифицирует работу с игровыми ресурсами (текстуры, модели, музыка и т.д.). Все игровые ресурсы помещаются в единое хранилище, а их обработкой занимаются заранее подготовленные импортеры. Таким образом, разработчику не нужно тратить свое время на создание классов, которые бы загружали ресурсы в игру.

Автор – Иван Андреев andreev.ia@gmail.com

Руководство по публикации приложений в Windows Phone Marketplace

Рейтинг статьи

Зарегистрировавшись на портале App Hub, вы сможете отправлять свои игры и приложения для Windows Phone в магазин Windows Phone Marketplace. Магазин Marketplace доступен с любого устройства Windows Phone 7 и позволяет пользователям покупать и загружать приложения прямо на свой телефон. Обработка ваших платежей выполняется автоматически. Активируйте для своего приложения ознакомительный режим, чтобы потенциальные покупатели могли опробовать его перед приобретением.

Перед отправкой своего проекта (приложения или игры) внимательно ознакомьтесь с Правилами сертификации приложений.

Если вы готовы, войдите на портал App Hub, откройте свою панель мониторинга и инициируйте отправку нового приложения. Загрузите файл XAP на сервер и введите метаданные: описание, категорию и изображения. Выберите цены и страны, в которых будет производиться продажа. В ходе отправки метаданных производится проверка файла XAP; в случае успешного прохождения проверки вы сможете опубликовать файл сразу после сертификации или в другое более удобное время. Если проверка файла XAP завершится неудачно, то вам будут сообщены причины.

Проверенные файлы XAP переупаковываются для автоматического и ручного тестирования на реальном телефоне в соответствии с Правилами сертификации приложений. Если приложение или игра соответствует всем сертификационным требованиям, то перепакованные файлы XAP и файлы сборки подписываются корпорацией Майкрософт и считаются официально сертифицированными. После этого их можно публиковать в магазине Windows Phone Marketplace. В случае неудачной сертификации файла или игры вам будет предоставлен соответствующий отчет.

Следующая >>>

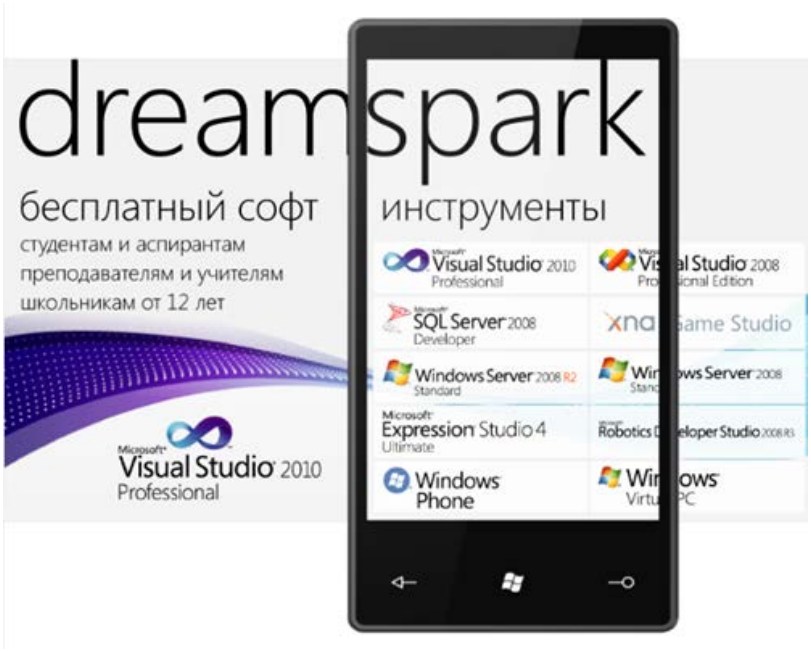
Публикация приложений

Рекомендуемые книги

Программируем Windows Phone 7.
(Полное издание)

Руководство на Marketplace для студентов

Как вы, наверняка, уже знаете, для участников программы DreamSpark (зарегистрироваться могут студенты, аспиранты, школьники >12 лет и преподаватели) открыта возможность бесплатно размещать приложения в App Hub с тем, чтобы дальше они могли попасть в Windows Phone Marketplace (для обычных разработчиков доступ стоит $100).



Ниже небольшая пошаговая инструкция о том, что нужно сделать.

Регистрация в DreamSpark

Для регистрации в DreamSpark вы можете воспользоваться любым из способов, перечисленных на странице http://dreamspark.ru/access.aspx:

- 25-значный код доступа, который можно получить в рамках тех или иных акций (например, на наших мероприятиях)
- Карта ISIC (ITIC)
- Email в домене учебного учреждения (например, в рамках программы Live@edu)
- Предъявив скан документа, подтверждающего академический статус
- Для школьников: регистрация через преподавателя.

В зависимости от способа далее последовательность действий может немного различаться, но общий сценарий таков:

1. Зайти на DreamSpark.com под своим LiveId
2. Пройти верификацию (Get Verified) с кодом, номером карточки или email.

Установка инструментов

После регистрации в DreamSpark, необходимо установить инструменты для разработки приложений под Windows Phone (до проверки личности в GeoTrust в любом случае нужно попробовать загрузить хотя бы одно минимальное приложение).

Минимальный набор:

- Windows Phone Developer Tools

Рекомендуемый набор:

Публикация приложений

Рекомендуемые книги

Программируем Windows Phone 7. (Полное издание)

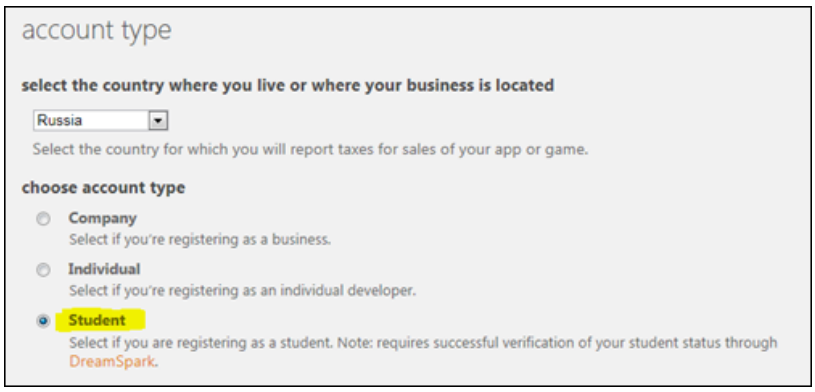
- Visual Studio 2010 (Express или Professional)
- Expression Studio 4 (достаточно из пакета поставить Expression Blend)
- XNA Game Studio 4
- Обновления к этим инструментам.
- Windows Phone Developer Tools – ставится поверх Visual Studio и Expression Blend.

Если вы хотите разрабатывать на Visual Basic, необходимо также установить Visual Basic for Windows Phone Developer Tools.

Регистрация в App Hub

App Hub – это специальный ресурс для размещения своих приложений и проверки их текущего статуса, включая статистику загрузок.

Для регистрации перейдите к созданию нового профайла, убедитесь, что вы регистрируетесь как студент:



(также необходимо регистрироваться с тем же LiveId, который вы используете в DreamSpark).

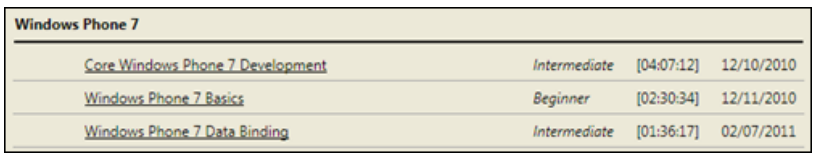
Закончите процесс регистрации, указав запрашиваемые данные.

Создание первого приложения

Для подтверждения своей личности через GeoTrust вам необходимо загрузить в App Hub любое, даже самое простое, приложение, например, то, которое сразу создает Visual Studio и базового шаблона. Конечно, вы можете сразу попробовать опубликовать и финальную версию вашего приложения, которое вы хотите распространять бесплатно или за деньги.

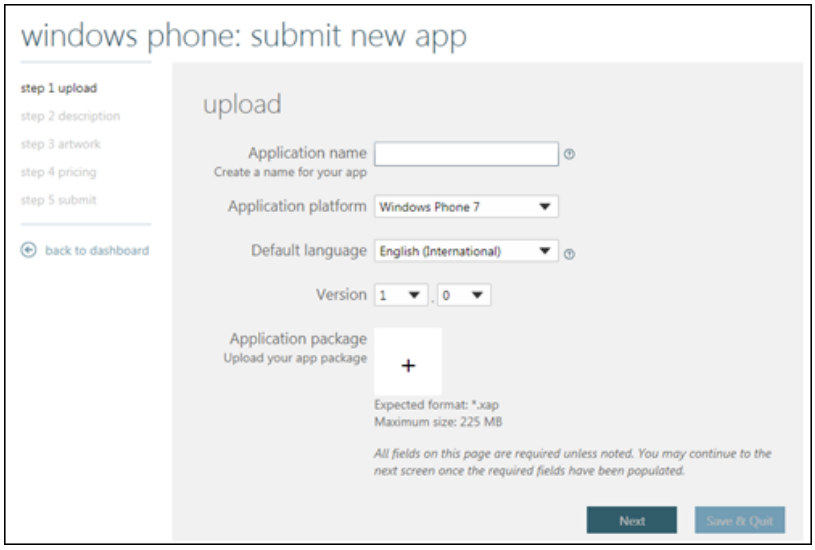
Чтобы вам было проще научиться разрабатывать под Windows Phone, мы ранее опубликовали большую подборку полезных материалов: Windows Phone 7. Куда пойти учиться?

Также участникам DreamSpark доступны тренинги PluralSight, среди которых есть три тренинга непосредственно по Windows Phone:



Верификация в GeoTrust

Если вы готовы загрузить свое первое приложение, перейдите в App Hub в раздел загрузки приложения и заполните необходимые данные:



Далее в течение 24 часов должно прийти письмо от GeoTrust (если не пришло, не забудьте проверить спам-фильтры).

GeoTrust попросит вас подтвердить вашу личность сканом какого-либо документа, удостоверяющего, что вы – это вы (например, паспорт или водительские права).

После подтверждения ваших данных, анализ вашего приложения длится до 5 рабочих дней. После подтверждения вашего приложения, оно должно появиться в Marketplace в течение 1 рабочего дня.

Дополнительные рекомендации

Для ускорения процесса регистрации и верификации через GeoTrust можно использовать следующие "лайв-хаки":

- *Загрузите первым делом самое простое приложение, которое наверняка не пройдет дальнейшую верификацию. Например, приложение без иконки.*
- *При подаче заявки укажите любую стоимость приложения, не равную 0, чтобы это не влияло на количество бесплатных для пользователей приложений, которые вы планируете загрузить (максимум 5).*

Как только Microsoft получит заявку вашего приложения, даже если оно не пройдет процесс тестирования, паралельно будет запущен процесс верификации вашей личности через GeoTrust.

Видео-инструкция: см. также видео-инструкцию по процессу регистрации – скринкаст от Rob Miles.

Разблокировка телефона для разработки

Если у вас есть физический телефон на базе Windows Phone 7, сразу после прохождения верификации вы можете его разблокировать. Для этого:

1. В меню Пуск выберите Windows Phone Developer Tools
2. Откройте Windows Phone Developer Registration
3. Подключите телефон, должен запуститься Zune-клиент, утилита для регистрации должна сказать, что все готово для разблокировки
4. Введите свои данные с App Hub и нажмите Register.

Дальше из App Hub вы можете также управлять количеством разблокированных телефонов.

Мастер-класс по дизайну приложений для Windows Phone

Рейтинг

Часть 1

Часть 2

Обучающие материалы

Основы Windows Phone 7

- День #1. Общие знания о проекте
- День #2. Цвета и цветовые темы
- День #3. Ориентация экрана
- День #4. Кнопка "Назад"
- День #5. Навигация
- День #6. Общие данные, передача данных в приложении
- День #7. Некромантия. Или tombstoning в WP7
- День #8. Изолированное хранилище
- День #9. Application Bar
- День #10. Немного о панораме. Программный контроль
- День #11. InputScope или какие бывают клавиатуры
- День #12. Создание триальной версии приложения
- День #13. Задачи выбора и запуска

Windows Phone 7

- Нелинейная навигация и петли в Windows Phone 7
- Использование памяти и немного о производительности
- Расширение базовой функциональности элементов управления на примере HyperlinkButton
- Как запустить несколько эмуляторов
- Работа с акселерометром
- Работа с GPS
- Следим за ошибками
- Правильный вывод веб сайтов на WP7
- Starter Kits
- Учимся программировать игры на XNA для Windows Phone 7 «Mango» — начало
- Дизайн приложений для WP7. Metro-подход
- Мастер-класс по дизайну приложений для Windows Phone
- Push Notification. Необходимость
- Push Notification. Как это работает
- Push Notification. Модель и действительность
- Рисование. Фигуры и линии, манипуляции с объектами

Советы

- Стартовый и загрузочный экран - это первое впечатление... Сделайте его отличным!
- Убедитесь что кнопки видны даже тогда, когда показана клавиатура
- Стройте свое приложение с учетом того, что существуют темы и "акцентируемые" цвета
- Убедитесь, что палец попадает по цели и что текст

Описание

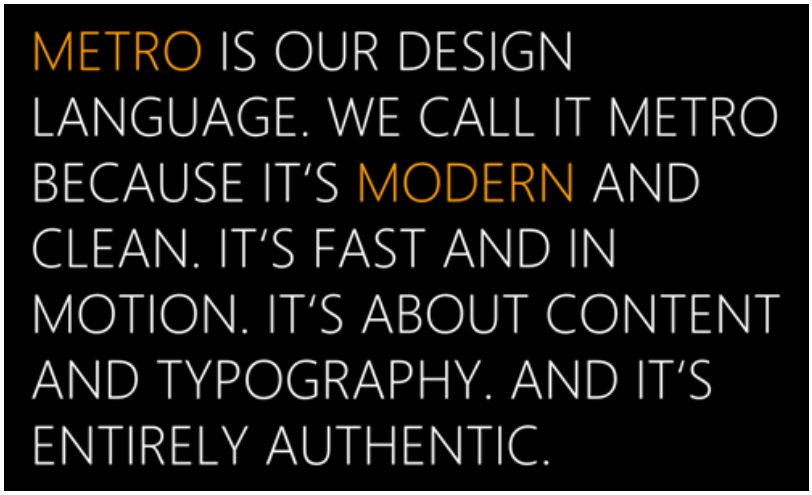
Мастер-класс Константина Кичинского по дизайну приложений для Windows Phone.

Автор: Константин Кичинский
Продолжительность:1 часть: 110 минут 05 секунд; **2 часть**: 124 минуты 24 секунды

Дизайн приложений для WP7. Metro-подход

Рейтинг

Если вы видели Windows Phone 7, вы уже видели Metro. Metro — это язык дизайна для приложений, взрощенный в недрах Microsoft, элементы которого уже проникают в разные продукты и, безусловно, это душа платформы WP7. Metro — это старт с нуля, ресет дизайна, переход от трудно поддерживаемого языка Windows Mobile к языку с четкими принципами и задачами.



Когда несколько лет назад команда дизайна решила попробовать начать с чистого листа, вместо того, чтобы смотреть на то, что уже есть на различных, в общем-то, однообразных платформах, она сконцентрировалась на том, что действительно вдохновляет — лучших образцах дизайна: от Josef Müller-Brockmann (швейцарский дизайнер, известный своим простым дизайном с ярким использованием типографики, формы и цвета, вдохновивший своими работами многих современных графических дизайнеров) и других пионеров International Style, дизайнерской системы Massimo Vignelli карты нью-йоркского метро и известных брендов вроде American Airlines до концептуальных работ Experimental Jetset.

Схожие источники вдохновления использовались при дизайне Windows Media Center, Zune и Xbox — и в этом смысле это продукты одной дизайнерской мысли.

Обучающие материалы

Основы Windows Phone 7

- День #1. Общие знания о проекте
- День #2. Цвета и цветовые темы
- День #3. Ориентация экрана
- День #4. Кнопка "Назад"
- День #5. Навигация
- День #6. Общие данные, передача данных в приложении
- День #7. Некромантия. Или tombstoning в WP7
- День #8. Изолированное хранилище
- День #9. Application Bar
- День #10. Немного о панораме. Программный контроль
- День #11. InputScope или какие бывают клавиатуры
- День #12. Создание триальной версии приложения
- День #13. Задачи выбора и запуска

Windows Phone 7

- Нелинейная навигация и петли в Windows Phone 7
- Использование памяти и немного о производительности
- Расширение базовой функциональности элементов управления на примере HyperlinkButton
- Как запустить несколько эмуляторов
- Работа с акселерометром
- Работа с GPS
- Следим за ошибками
- Правильный вывод веб сайтов на WP7
- Starter Kits
- Учимся программировать игры на XNA для Windows Phone 7 «Mango» — начало
- Дизайн приложений для WP7. Metro-подход
- Мастер-класс по дизайну приложений для Windows Phone
- Push Notification. Необходимость
- Push Notification. Как это работает
- Push Notification. Модель и действительность
- Рисование. Фигуры и линии, манипуляции с объектами

Советы

- Стартовый и загрузочный экран - это первое впечатление... Сделайте его отличным!
- Убедитесь что кнопки видны даже тогда, когда показана клавиатура

Стройте свое приложение с учетом того, что

Команда, занимающаяся дизайном Windows Phone пошла дальше и разработала набор принципов для направления в правильное русло интерактивного дизайна, дизайна движения и общих впечатлений от работы с телефоном.

(Честно говоря, мне трудно было дословно перевести все принципы, но я постараюсь сохранить содержательную сущность.)

Принципы metro-дизайна

Легкий, простой, открытый и быстрый

Этот принцип можно было бы назвать тотальной зачисткой (Fierce Reduction), мы стремимся убрать из UI все элементы, которые кажутся ненужными; причем как визуальные элементы, так и переизбыток функциональности. Это побуждает сфокусироваться на первоочередных задачах UI и позволяет UI выглядеть умным, открытым, быстрым и отзывчивым.

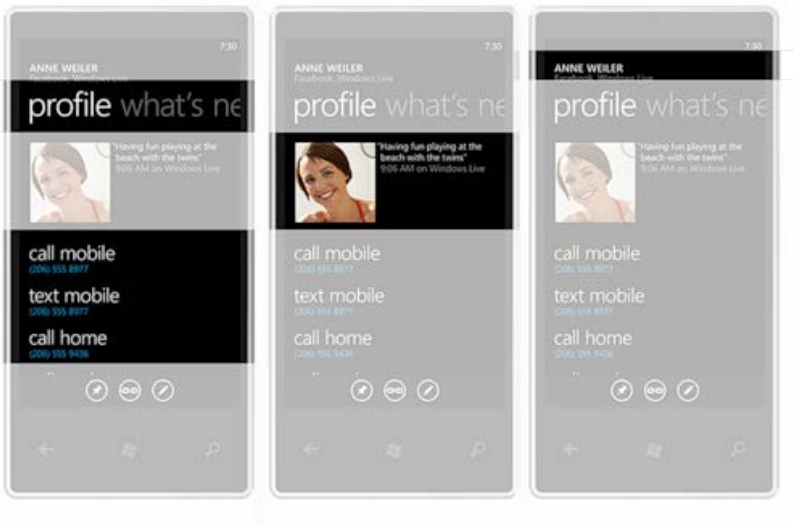
UI должен позволить пользователю быстро схватить самую суть информации и только при необходимости погрузиться в детали. И вместе с тем, предложение к погружению и исследованию тоже можеть быть частью UI, именно эту цель, например, решает панорама в Windows Phone.

Приоритезируйте, выделив основные задачи, старайтесь достигать большего меньшими средствами и убирать все лишнее. И, конечно, свободное пространство — тоже один из важных компонентов проектируемого UI. Пустого пространства не надо бояться — надо уметь с ним работать. Заполнить все имеющееся пространство чем-либо не должно быть самоцелью. Многие вещи можно упростить, показать схематично или заведомо символично.

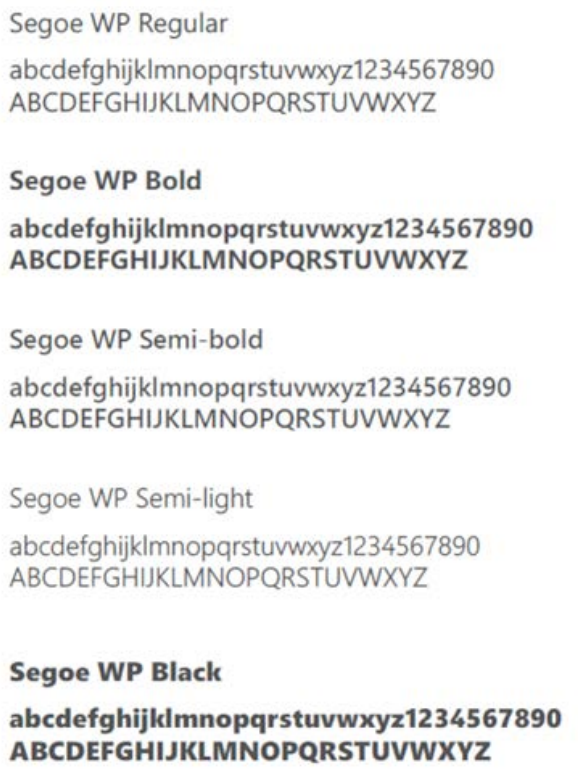
Продумывайте функционал вашего приложения и выстраивайте его в соответствии с важностью и частотой использования, выставляя на первое место именно то, зачем пришел пользователь, и убирая все остальное на второй план.

Качественная типографика

Metro-дизайн в значительной степени инспирирован лучшими образцами качественной типографики. Похоже, пришло время пользовательскому интерфейсу также быть завязанным на типографику. Шрифт — это информация. Красота, четкость и правильность выбора шрифта — путь к открытому и понятному информационному дизайну.



Думая об использовании шрифта, помните о толщине, размере, регистре и балансе различных элементов UI. В частности в разных функциональных блоках может использоваться написание текста строчными, прописными буквами или же как в предложении. В Windows Phone иерархия навигации выстраивается посредством соответствующего /физического/ размещения текстовых блоков относительно друг друга и подчеркивается размером и регистром шрифта. Например, заголовок раздела или приложения в Pivot пишется меньшим шрифтом и в верхнем регистре, а заголовок текущей вкладки крупнее и в нижнем регистре.



Для платформы WP7 была разработана модификация семейства шрифтов Segoe — Segoe WP, доступного, как в инструментах дизайна и разработки, так и на самом устройстве.

Живой в движении

Metro — это живая система, в которой движение, отзывчивость и реакция, переходы также важны, как и экраны, с которыми взаимодействует пользователь. Движение придает характер UI и в то же время помогает разобраться в навигационной системе, что помогает улучшить юзабилити.

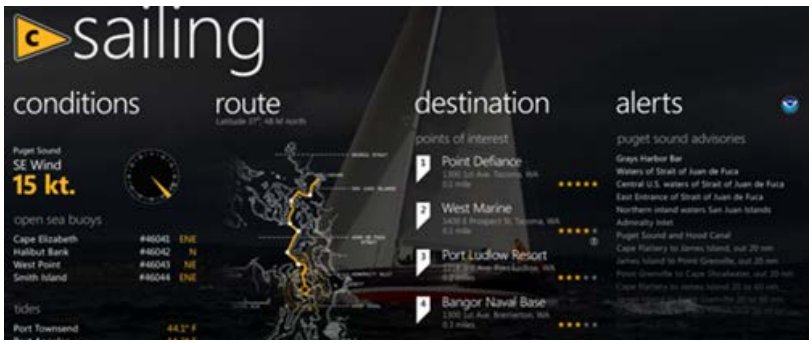
Движение дает ощущение глубины и погружения в контент, переход к деталям или назад к верхнему уровню. Джижение опирается на чувства размера и ориентации в пространстве. Живая система реагирует на действия пользователя, подсказывает, куда можно нажать и с чем можно взаимодействовать. На интуитивном уровне направляет и помогает разобраться, что к чему.

Отзывчивые элементы UI наполняют жизнью взаимодействие с системой. См. также множество видео с обзорами на канале Windows Phone на Youtube.

Контент прежде всего

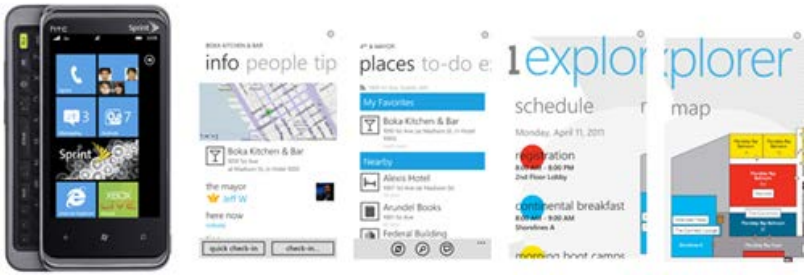
Пользователь приходит за информацией, а не нажимать на кнопки. Уменьшение визуальной составляющей, не являющейся контентом, поможет вам создать открытый и легкий UI. Информация должна располагаться так, чтобы побуждать пользователя ее исследовать и взаимодействовать с ней напрямую, а не через специальные кнопки. Важная информация должна преподноситься сразу, вторичная и детальная уходить на второй план, но быть доступной в одно действие.

Контент не должен быть просто сам по себе, он является частью частью UI — и даже больше: контент и есть UI.

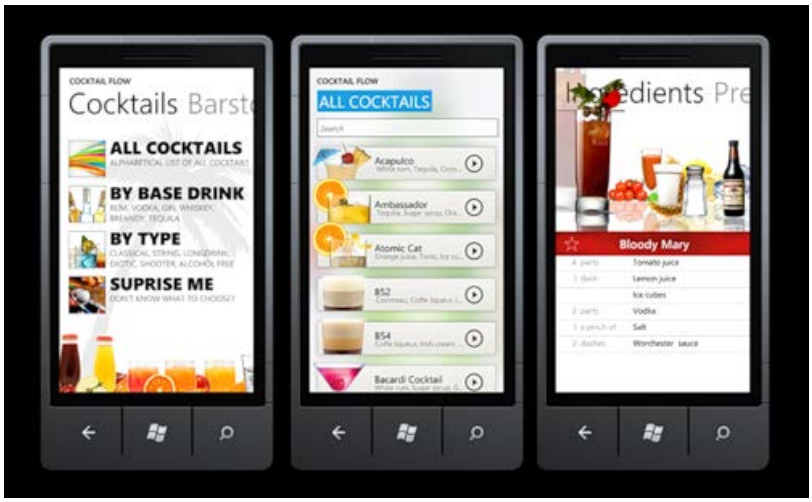


И в отличие от традиционных иконкографичных систем:

системы с metro-дизайном больше ориентированы на инфографику, в которой даже элементы на стартовом экране наполнены жизнью и информацией, а во внутреннем содержании приоритет отдан контенту, а не оформлению и кнопкам ;)



Если первый подход наполнен реалистичными метафорами из реального мира — этаким гиперреализмом в дизайне, в котором аналоговый контент преображается в цифровые аналоги и работа строится вокруг организации и манипуляциями над контентом, то второй стремится предоставить контент в том виде, как он есть, помня о том, что цифровая форма может давать дополнительные преимущества и возможности, и не всегда прямая аналогия — самый лучший способ. Инфографичный подход расширяет объекты (например, людей и места) релевантной имеющейся информацией, позволяет накладывать информацию и напрямую с ней взаимодействовать.



Это не означает, что фотографический (медиа) контент теряет актуальность в metro-дизайне. Напротив, он только приобретает важность, так как контент имеет абсолютный приоритет над элементами UI. Фотография — это контент, а не подложка для кнопки.

Цифровая честность

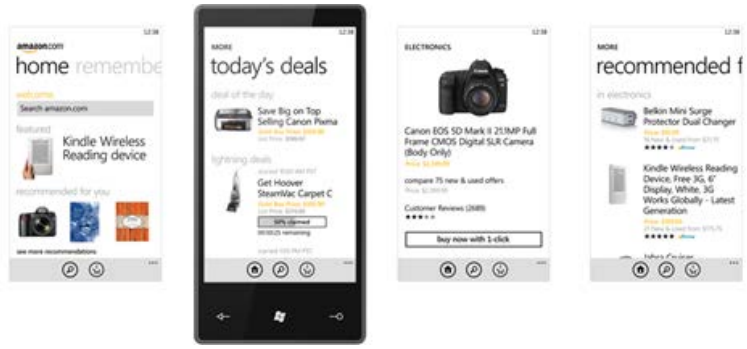
Наконец, metro-дизайн исповедует честность в дизайне. UI создается для пикселей, поэтому в Metro мы стараемся избежать использования аналогий с реальным миром в виде теней и бликов, используемых в некоторых UI для мимикрии под материалы и объекты реального мира (т.н. skeuomorphic-дизайн).

Честность предполагает также и дизайн под форм-фактор, учет особенностей взаимодействия с устройством с помощью пальцев рук и, конечно же, прямое изложение информации.

UI не должен быть тем, чем он не является. Будьте честными со своими пользователями.

Не всегда прямой перенос решения с другой платформы на WP7 без учета особенностей дизайна будет самым лучшим решением. Попробуйте встроиться в платформу и в те особенности дизайна, которой она живет.

Заставьте metro работать на себя.



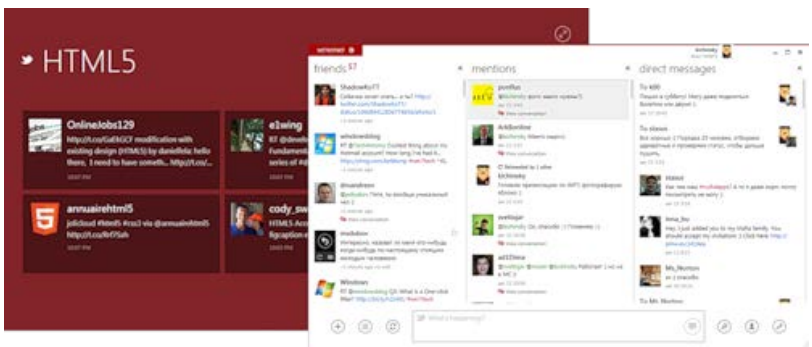
Настоящее и будущее

Хотя переход от Windows Mobile 6.5 к Windows Phone 7 — это сам по себе большой шаг, команда дизайна, занимающаяся Metro, смотрит также и в будущее, предполагая осторожную и продуманную эволюцию. Metro не есть что-то, спроектированное, чтобы быть отличным от всего остального. Это фундамент для будущего развития на протяжении долгого времени, начальная точка на пути туда, что нам кажется следующей эрой в дизайне пользовательских интерфейсов, сфокусированном более на контенте, нежели на метафорах, информации, а не инструментах, и движении, а не статике. Это язык, спроектированный, чтобы четко обозначить информацию вокруг нас, убрав сопровождающий мусор.

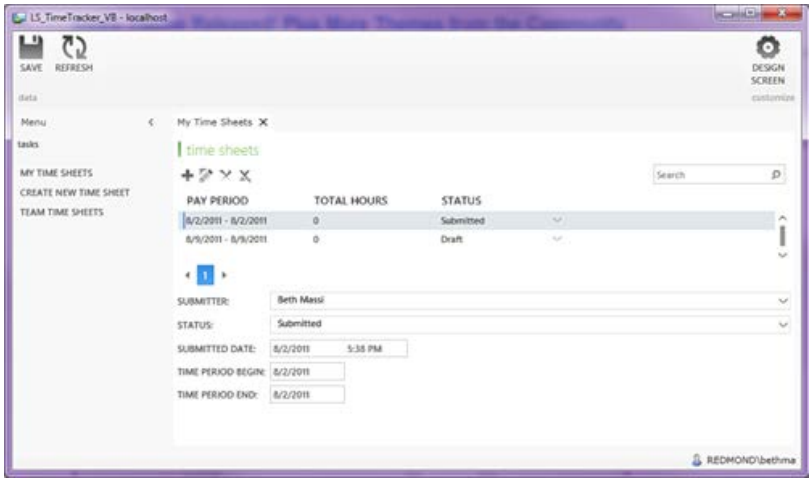
Интерфейсы вроде тех, что мы все видели в научной фантастике вроде Аватара или Железного человека 2 — это дело нескольких лет, но Metro уже сегодня кажется хорошей стартовой точкой.

Metro-подход в реальных проектах

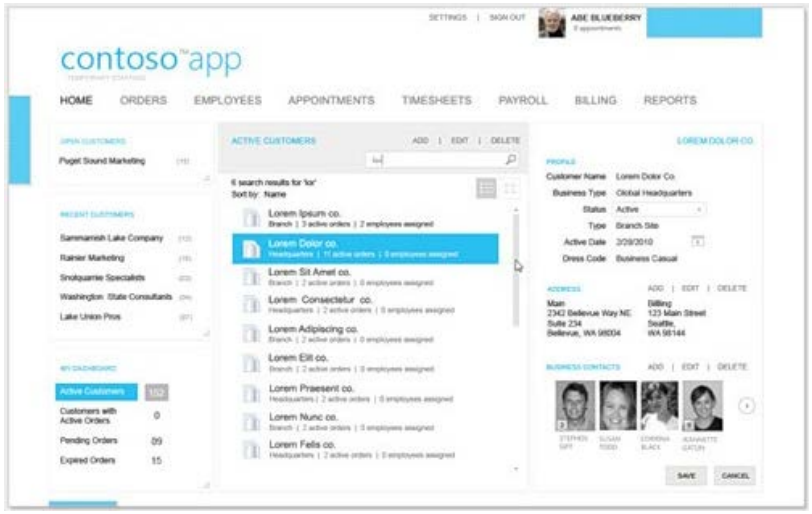
Идеи metro-дизайна уже во всю применяются в различных проектах (это даже не беря в рассчет тысячи приложений под Windows Phone 7).



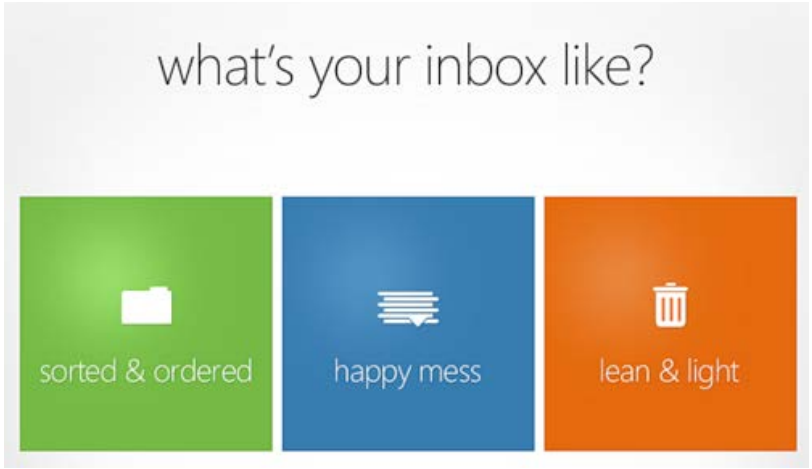
Уже сегодня вы можете попробовать MetroTwit — клиент для Twitter, сделанный в metro-стиле, или MetroTwit Show — интересное приложение для демонстрации потока твитов (например, во время конференции или просто в качестве экранной заставки).



Специальная metro-тема есть для приложений, сделанных с помощью Visual Studio LightSwitch.



И давно существует своя metro-тема для любых приложений на Silverlight и WPF. А многие наши партнеры (например, telerik), выпускающие контролы для платформы Microsoft, также трудятся над тем, чтобы они были в стиле Metro UI. Вот еще один пример metro в Silverlight от Alex Knight.



Еще один интересный пример — сайт про Windows Live Hotmail. Кстати, в нем используется HTML5.

Наконец, вы наверняка уже видели наш обновленный сайт MSDN или сайт P&P Summit. Совершенно верно, metro-проникает в сайто-строение не меньше, чем на десктопе или мобильные платформы.

И, хотя, в первую очередь, во всех этих случаях в глаза бросается узнаваемость стилистики, самое важное — это чтобы за использованием тех или иных графических элементов не терялась суть, то есть те принципы, о которых мы говорили с самого начала:

- Легкий, простой, открытый и быстрый
- Качественная типографика
- Живой в движении
- Контент прежде всего
- Цифровая честность

Windows 8

Хотя мы не можем рассказать каких-либо дополнительных деталей сегодня, вы, наверняка, уже видели обзор интерфейса следующей версии Windows. В блоге Building Windows 8 (кстати, есть версия на русском) вы сможете найти некоторые дополнительные детали в статье Designing for Metro style and the desktop.

Полезные ссылки

- Application Design for Windows Phone by Megan Donahue
- .toolbox design trainings
- Windows Phone Design Day Recordings
- WP7 UI Guide & Design Templates
- User Experience Design Guidelines for Windows Phone
- Книга Designing for Windows Phone
- Обновленные бесплатные инструменты для разработки и дизайна — Windows Phone SDK 7.1 RC, включающие Microsoft Expression Blend SDK for Windows Phone 7.1

См. также мой доклад на предыдущем BizSpark Camp по Windows Phone.

45 дней с Windows Phone 7. День #1. Шаблоны проекта.

Рейтинг статьи

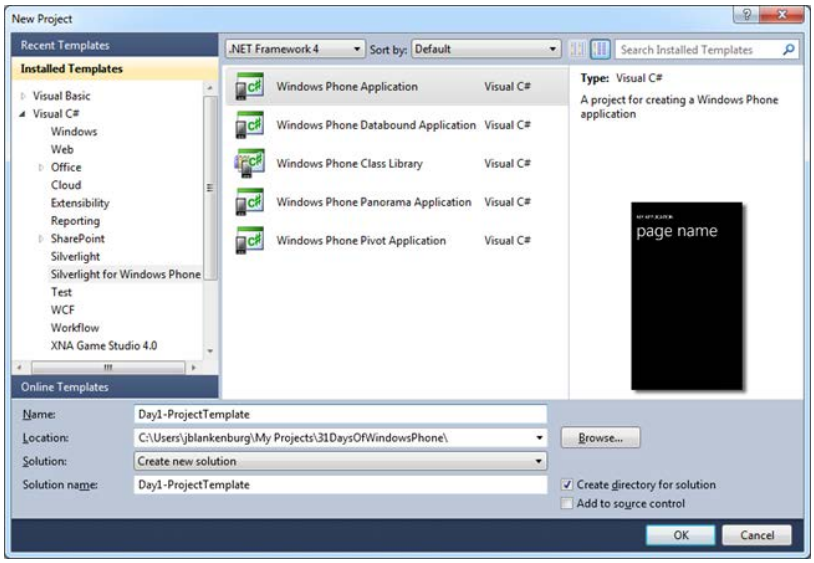
Данная серия основывается на англоязычной серии статей «31 Days of Windows Phone 7» от Jeff Blankenburg, но не просто копирует её, а расширяет и дополняет.

Для понимания статей данной серии требуется некоторое базовое знание Silverlight (в изучении данной технологии Вам могут помочь доклады на сайте TechDays.ru). Кроме того, хотя в некоторых статьях будет обсуждаться создание игр для телефонов, данная тема рассматривается с нуля.

Для работы всех примеров предполагается, что у Вас установлены инструменты разработки под Windows Phone 7. Скачать их можно с сайта http://developer.windowsphone.com, пройдя по ссылке «Get the free tools», что говорит нам об их абсолютной бесплатности.

После установки инструментов Вам будут доступны Visual Studio 2010 for Windows Phone и Expression Blend 4 for Windows Phone. Для наиболее удобного создания Windows Phone 7 приложений Вам понадобятся оба этих продукта. Если же у Вас на компьютере установлены более старшие версии Visual Studio 2010 или Expression Blend 4, то вместо установки отдельных продуктов, в существующих приложениях просто появятся новые типы проектов, необходимые для разработки под телефон.

Ну что же, давайте посмотрим из чего состоит стандартный шаблон проекта Windows Phone 7 приложения. В Visual Studio в меню File выберем New -> Project... и в открывшемся диалоговом окне перейдём в раздел Silverlight for Windows Phone. Здесь представлено несколько типов проектов, о которых мы поговорим позднее. Сейчас же рассмотрим базовый тип проекта: Windows Phone Application. Создадим новый проект данного типа.

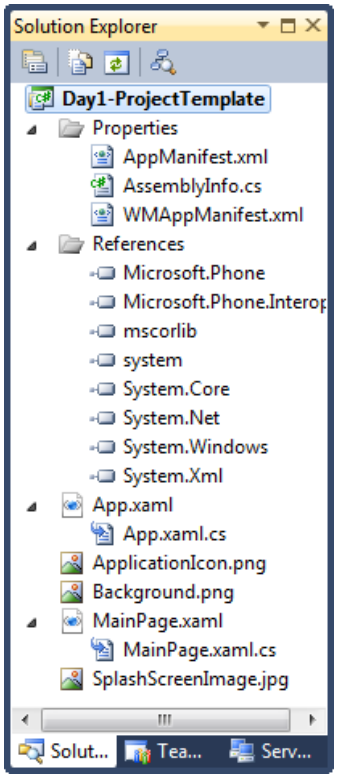


Solution Explorer

После создания проекта в окне Solution Explorer мы можем посмотреть на его структуру. В данный момент мы не будем рассматривать содержимое каждого файла, а обсудим то, для чего они нужны. Если Вам хочется увидеть содержимое файлов, просто установите инструменты, о которых мы говорили в начале статьи, и начните разработку Windows Phone 7 приложений.

Надо отметить, что многие файлы настроек приложения обычно редактируются не напрямую, а в специальном графическом интерфейсе.

Статьи данной серии



ApplicationIcon.png

Картинка, которая будет иконкой Вашего приложения в телефоне. Это действительно важный файл, так как он является первым, что увидят пользователи при работе с Вашим приложением.

App.xaml

Хотя данный файл и не является полным аналогом, он служит целям, напоминающие таковые у файла web.config в ASP.NET приложениях. App.xaml является тем местом, где Вы можете хранить данные и настройки для всего приложения. Также в данном файле удобно хранить стили, но это не является обязательным.

App.xaml.cs

Файл кода (code-behind) для App.xaml. Здесь Вы можете обрабатывать события и ошибки уровня приложения, в том числе его «захоранивание»(«tombstoning»). Данную концепцию мы рассмотрим позднее, когда будем говорить про многозадачность.

AppManifest.xml

Простой манифест, необходимый для генерации XAP файла (пакета приложения).

AssemblyInfo.cs

Ещё один конфигурационный файл, в котором определяются некоторые метаданные главной сборки(Assembly) приложения.

Background.png

Данная картинка используется, когда Ваше приложение закреплено на стартовом экране телефона(start screen). По сути это большая иконка приложения. Поэтому было бы разумно сделать её визуально похожей на ApplicationIcon.png.

MainPage.xaml

Первая страница Вашего приложения. Именно её увидит пользователь после собственно загрузки приложения. Почти во всех приложениях данная страница не будет и не должна быть единственной. Телефон очень хорошо обрабатывает навигацию вперёд/назад, и было бы не разумно запихивать всю функциональность приложения в одну страницу. Тему навигации мы рассмотрим совсем скоро, а именно в следующей статье.

MainPage.xaml.cs

Файл кода страницы MainPage.xaml. Когда Вам требуется написать какой-либо код для страницы, Вы можете использовать MainPage.xaml.cs в этих целях. Однако, существуют и другие подходы, например паттерн проектирования MVVM (Model-View-ViewModel), когда файл кода страницы оказывается почти пустым, а бизнес логика определяется в других классах. Но так как писать код для небольших примеров непосредственно в *.xaml.cs файлах удобнее в дальнейшем мы выберем именно такой подход, хотя

паттерны проектирования также рассмотрим.

SplashScreenImage.jpg

Данная картинка отображается во время загрузки Вашего приложения. Вы можете задать свою картинку, но учтите, что её целью является только информирование пользователя о том, что приложение загружается. В этом месте не стоит пытаться сделать что-то необычное.

WMAppManifest.xml

Файл метаданных, который содержит множество настроек приложения: заголовок, задание первой страницы, пути к иконкам, определение необходимых системных возможностей и.т.д.

Автор Jeff Blankenburg
Перевод и доработка Сергей Пугачёв

Основы Windows Phone 7

Рейтинг статьи

Добро пожаловать на Основы Windows Phone 7

Сегодня мы с Вами поговорим о том, как создать свое приложение на платформе Windows Phone 7. Каждая из этих статей, покрывает небольшой кусочек платформы, не углубляясь в детали, что бы вы смогли почувствовать себя если не уверенно, то хоть твердо на двух ногах при разработке своего приложения. Итак, вот список тем, необходимых для начала:

- **День #1. Общие знания о проекте**
- **День #2. Цвета и цветовые темы**
- **День #3. Ориентация экрана**
- **День #4. Кнопка "Назад"**
- **День #5. Навигация**
- **День #6. Общие данные, передача данных в приложении**
- **День #7. Некромантия. Или tombstoning в WP7**
- **День #8. Изолированное хранилище**
- **День #9. Application Bar**
- **День #10. Немного о панораме. Программный контроль**
- **День #11. InputScope или какие бывают клавиатуры**
- **День #12. Создание триальной версии приложения**
- **День #13. Задачи выбора и запуска**

Автор Антон Полховский

Comments (0)

Leave a Comment
Sign in to leave a comment.

Обучающие материалы

Основы Windows Phone 7

- День #1. Общие знания о проекте
- День #2. Цвета и цветовые темы
- День #3. Ориентация экрана
- День #4. Кнопка "Назад"
- День #5. Навигация
- День #6. Общие данные, передача данных в приложении
- День #7. Некромантия. Или tombstoning в WP7
- День #8. Изолированное хранилище
- День #9. Application Bar
- День #10. Немного о панораме. Программный контроль
- День #11. InputScope или какие бывают клавиатуры
- День #12. Создание триальной версии приложения
- День #13. Задачи выбора и запуска

Windows Phone 7

- Нелинейная навигация и петли в Windows Phone 7
- Использование памяти и немного о производительности
- Расширение базовой функциональности элементов управления на примере HyperlinkButton
- Как запустить несколько эмуляторов
- Работа с акселерометром
- Работа с GPS
- Следим за ошибками
- Правильный вывод веб сайтов на WP7
- Starter Kits
- Push Notification. Необходимость
- Push Notification. Как это работает
- Push Notification. Модель и действительность
- Рисование. Фигуры и линии, манипуляции с объектами

Советы

- Стартовый и загрузочный экран - это первое впечатление... Сделайте его отличным!
- Убедитесь что кнопки видны даже тогда, когда показана клавиатура
- Стройте свое приложение с учетом того, что существуют темы и "акцентируемые" цвета
- Убедитесь, что палец попадает по цели и что текст можно прочитать
- Показывайте прогресс и отзывы на нажатие
- Встройка веб контента должна выполняться с

- особой осторожностью
- Располагайте кнопки правильно... Летающие кнопки, кнопки Домой, кнопки назад...!
- Понимание Pivot и Panorama
- Облегчите себе жизнь. Используйте обычные контролы, но используйте их правильно!
- Расположение элементов - действительно имеет значение!

Дополнительные материалы

- Создание приложений для Windows Phone с использованием Silverlight+XNA
- Интеграция приложения на Windows Phone со SkyDrive
- Использование HTML5 и JavaScript для разработки приложений под Windows Phone
- Переключение слайдов презентации с использованием Windows Phone: UDP соединение
- Переключение слайдов презентации с использованием Windows Phone: TCP соединение
- Дизайн приложений для WP7. Темы и акценты
- Прокрустовы окна. Как вписаться в устройства с минимальными потерями (видео)
- HTML5 для разработки мобильных приложений (видео)
- appmakr: Как сделать приложение для Windows Phone за 10 минут и без программирования
- Разработка игр для Windows Phone (видео)
- Пример простого онлайн-проигрывателя
- Пару слов о Motion API
- Монетизация приложений — где деньги лежат, или почему это все реклама?
- Работа с HTTP в 3 строчки, или делаем свой RSS Reader с WebClient и WebBrowser
- Быстрое создание контент-оринетированых приложений
- Карты и гелокационные данные на Windows Phone
- Обновления Live Tiles в фоновых агентах
- Всё о Splash Screen в Windows Phone

Полезные ресурсы

- Хаб по Metro-дизайну
- Разработка 3D игр
- Центральный ресурс для разработчиков под Windows Phone 7
- Руководство по публикации приложений в Marketplace
- Руководство по публикации приложений в Marketplace для студентов
- Группа разработчиков под Windows Phone 7 на Facebook
- Мастер-класс по дизайну приложений для Windows Phone
- Дизайн приложений для WP7. Metro-подход
- Серия статей "45 дней с Windows Phone 7
- Узнайте больше о Windows Phone 7
- Дополнительные материалы

Разработка приложений для Windows Phone 7

Это новая мобильная платформа, которая позволяет создавать интерактивные приложения и игры на Silverlight и XNA для пользователей по всему миру, объединенных сетевыми сервисами.

Учебный курс от новичка к эксперту

- Первое знакомство с платформой
- Шаблон проекта, страницы и навигация
- Стандартные варианты разметки
- Элементы управления Pivot и Panorama
- Основные элементы управления
- Текстовые поля и контекст ввода
- Задачи запуска (Launchers)
- Задачи выбора (Choosers)
- Элемент управления Map
- Элемент управления WebBrowser
- Работа с акселерометром
- Определение местоположения
- Работа с HTTP
- Работа с изолированным хранилищем
- Знакомство с локальной базой данных
- Жизненный цикл и сохранение состояния приложения
- Фоновые сервисы, запускаемые по расписанию
- Фоновая загрузка/выгрузка файлов
- Фоновое проигрывание музыки
- Оповещения
- Живые тайлы
- Push Notification

Новости

Visual Studio 11 и разработка приложений для Windows 8 вошли в число ключевых тем на конференции DevCon'12
2 марта в Москве прошла конференция Windows 8 Camp, во время которой разработчики познакомились с особенностями разработки Metro-приложений для Windows 8 с помощью Visual Studio 11, beta-версия которо... подробнее
понедельник, мар 5 **I.Vorontsov**

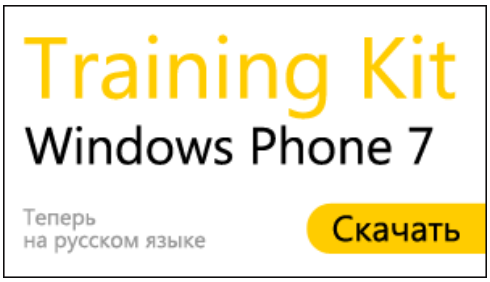
Новые материалы на Русском MSDN!
Уважаемые коллеги!Ниже приведена подборка подготовленных и опубликованных на MSDN материалов за последнии 10 дней:WebРепозиторий Helicon Zoo. Знакомство с GoliathWindows PhoneВсё о Splash Scre... подробнее
вторник, фев 28 **I.Vorontsov**

Опубликовано назнавние первых 100 докладов конференцииTechEd Europe 2012
Опубликовано назнавние первых 100 докладов крупнейшей европейской конференции TechEd Europe 2012, которая пройдет с 26 по 29 июня 2012 года в Амстердаме . В ходе конференции посетители смогут посетить... подробнее
вторник, фев 21 **I.Vorontsov**

Запущена Beta версия нового крупнейшего технологического видео хостинга TechDays!
Запущена beta версия нового технологического видео хостинга techdays! Если вы специалист в области разработки программного обеспечения или администрирования ИТ систем данный ресурс именно для вас. ... подробнее
понедельник, фев 20 **I.Vorontsov**

Все новые материалы Русского MSDN в одном агрегаторе
На Русском MSDN появилась возможность подписаться на RSS со всеми новыми материалами ресурса. Так же можно подписаться на RSS только по интересующему вас направлению в разработке, на данный момент дос... подробнее
четверг, фев 16 **I.Vorontsov**

Видео-доклады по Windows Phone 7 на русском языке



Новые материалы

Всё о Splash Screen в Windows Phone

Метро-дизайн. Навигация, уровни, назад и циклы

Метро-дизайн. Темы и акценты

Метро-дизайн. Контрастность и доступность

Метро-дизайн. Практика. Градиенты и 16-битность

Архив обновлений

Полезные ресурсы

- Хаб по Metro-дизайну
- Разработка 3D игр
- Центральный ресурс для разработчиков под Windows Phone 7
- Руководство по публикации приложений в Marketplace
- Руководство по публикации приложений в Marketplace для студентов
- Группа разработчиков под Windows Phone 7 на Facebook
- Мастер-класс по дизайну приложений для Windows Phone
- Дизайн приложений для WP7. Metro-подход
- Серия статей "45 дней с Windows Phone 7
- Узнайте больше о Windows Phone 7
- **Дополнительные материалы**

Блог

Промежуточные итоги конкурса - 23 февраля
В конце января мы совместно с Nokia запустили конкурс весенних приложений для Windows Phone. Конкурс... дополнительно
пятница, фев 24 **Mik Chernomordikov**

Промежуточные итоги конкурса - 14 февраля
В конце января мы совместно с Nokia запустили конкурс весенних приложений для Windows Phone. Конкурс... дополнительно
среда, фев 15 **Mik Chernomordikov**

Литература на зиму - новый WP7 Training Kit
Длинные новогодние выходные – хорошая возможность засесть за изучение технологий! Тем более, что не... дополнительно
вторник, янв 3 **Mik Chernomordikov**

Праздник в Marketplace приходит - результаты!
Месяц назад мы анонсировали совместный с Nokia конкурс новогодних приложений – и вот пришло вр... дополнительно
понедельник, дек 26 **Mik Chernomordikov**

Форум

Использование зарегистрированных торговых марок в приложении.
Можно ли использовать логотипы типа Adidas или KFC и т.п. в приложении? Если можно то есть ли ... дополнительно
понедельник, мар 12

Задай свой вопрос MVP - эксперту по Мобильной разработке (WP7 и XNA)
Здравствуйте.В данной теме на ваши вопросы будут отвечать эксперты в мобильной разработ... дополнительно
воскресенье, мар 11

windows mobile 6.5 & visual studio 2010 ultimate
Здравствуйте! У меня вот такой вопрос! Как создавать приложения под windows mobile 6.5 на C#? ... дополнительно
суббота, мар 10

Стилизации Button
День добрый.Стоит следующая задача. Нужно стилизовать кнопку. Есть две картинки, соответственно для... дополнительно
четверг, мар 8

Рекомендуемые книги

Программируем Windows Phone 7. (Полное издание)

Помощь в устранении неисправностей и поддержка

Если вы ищете ответ на вопрос, вы можете найти ресурс с помощью функции поиска, задать вопрос на форумах или обратиться за помощью в службу технической поддержки Microsoft. Если вы ищете загружаемый файл, перейдите на страницу загружаемых файлов.

1 Поиск в базе знаний Microsoft

Для поиска в базе знаний Microsoft введите код ошибки или искомое слово в поле ниже

Поиск

2 Задать вопрос на форумах

Выполните поиск ответа на ваш вопрос в форумах или задайте вопрос, если не удается найти готовый ответ.

- Мобильные устройства
- Microsoft Device Emulator (EN)
- .NET Compact Framework (EN)
- Visual Studio Smart Device Development – Visual Basic and C# Projects (EN)
- Visual Studio Smart Device Development - Native C++ Project (EN)

Все форумы Русского MSDN

3 Обратиться в Департамент стратегических технологий Microsoft Россия

Департамент стратегических технологий Microsoft Россия обеспечивает взаимодействие Microsoft с российскими организациями, формирующими индустрию разработки программного обеспечения. К ним относятся:

- университеты и технические вузы;
- компании - разработчики программного обеспечения;
- крупные предприятия, использующие стратегические технологии и платформы Microsoft;
- профессиональные сообщества разработчиков и специалистов в сфере информационных технологий.

Если у вас есть вопросы, пожелания, связанные с разработкой программного обеспечения, мероприятиями и программами Microsoft для разработчиков, инструментами разработки Microsoft, мы всегда готовы на них ответить по адресу devtlz@microsoft.com.

4 Информационный центр Microsoft Россия

Почтовый адрес:	Телефон	Факс
121614 Россия г. Москва, Крылатская улица, д. 17	+7 (495) 916-7171 в Москве +7 (800) 200-8001 для регионов РФ	Укажите имя и фамилию получателя. +7 (495) 641-1040

5 Обратиться за дополнительной помощью в Microsoft

Щелкните ссылку ниже, чтобы просмотреть варианты интерактивной поддержки, предлагаемой Microsoft.

- Связаться со специалистом службы технической поддержки Microsoft

Microsoft Connect

Отправить в Microsoft информацию об ошибках или отзывы о продуктах

Дополнительные ресурсы

- Выяснить срок поддержки вашего продукта
- Подробнее о вариантах поддержки Premier Support

Помощь экспертов

- Найти эксперта
- Связаться с MVP