Advanced About Page for Windows Phone apps

This article shows how to create an advanced "About" or "Information" page for a Windows Phone application that includes mechanisms for feedback, sharing and technical support.

Introduction

Applications should have an "About Page" which contains relevant information about the application and its author. Most applications will provide the following minimal information:

- Application name
- Application version
- Copyright
- Summary/overview of app
- Website, Support and Privacy Policy

Some applications need to show more details, including application history, external rules, credits, thanks etc.

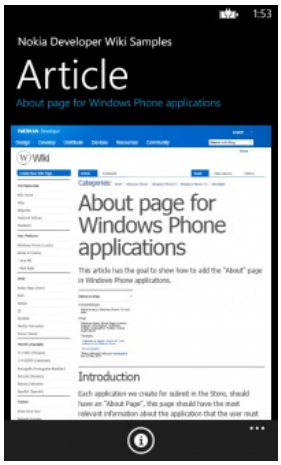
This article explains how create and reuse an advanced "About Page" which also includes functionality to:

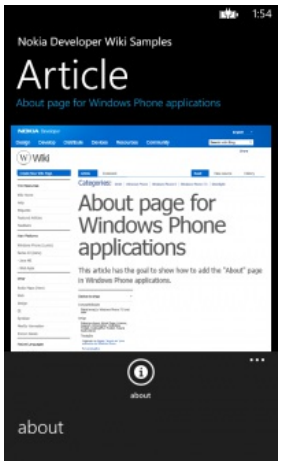
- Rate the application
- Send feedback to the author
- Share application by email
- Share application by social network

Screenshots

The first two screenshots below show the *About Page* with open and closed menu, respectively.

The following screenshots show the application home page with menu bar from which you can launch the About page. Note that it is the menu which is of interest here - the page itself is just a placeholder.





Note: The Home Page has both a button and menu item for "About" because in a real app it may be found in either place - here we show both implementations.

Source Code

The source code can be found here: About Page Sample

(C) Copyright Nokia 2014. All rights reserved.

Note: The Nokia Project will be discontinued and this project was moved for Github - NokiaProjects-About-Page-Sample

Pre-requisites

This example uses Model-View-ViewModel (MVVM), a design pattern recommended for Windows Phone application development.

There are a number of toolkits that can help make it easier to use the MVVM pattern. For this example we use:

- Cimbalino Windows Phone Toolkit - a set of useful and powerful items that will help you build your Silverlight applications for Windows Phone. The Toolkit is divided in projects which deliver different features. The base project (*Cimbalino.Phone.Toolkit*) contains base MVVM services, some very useful converters, helper classes and extension methods, and the bindable Application Bar behaviour.
- MVVM Light Toolkit (PCL version) - a set of components to help people use the MVVM pattern. It may be used in all XAML/C# frameworks.

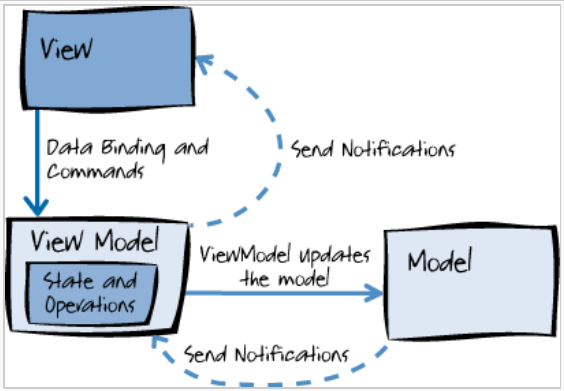


Diagram with an overview about MVVM pattern.

These packages are available in Nuget Package Manager (for both targets) and can be installed as described here:

- How to install MVVM Light Toolkit for Windows Phone
- How to install Cimbalino Windows Phone Toolkit packages

Note: For more information about MVVM see the following references:

- (MSDN) Implementing the MVVM Pattern
- (Pattern & Pratice) Developing a Windows Phone Application using the MVVM Pattern
- (Pattern & Pratice) Developing an Advanced Windows Phone 7.5 App that Connects to the Cloud
- (MSDN) Model-View-ViewModel (MVVM) Applications: General Introduction

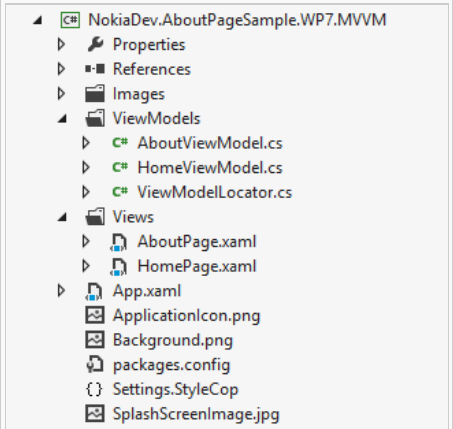
Creating an "About Page"

This code example is valid for both Windows Phone 7.5 and Windows Phone OS 8.0, and contains a complete sample for both platforms. The example app was created using the *Windows Phone App* template.

The project has the following files and folders:

- **ViewModels** - folder that will contain all view models
- **Views** - folder that will contain all views.
- **HomePage.xaml** - represents the main page
- **About Page.xaml** - represents the about page
- **HomeViewModel.cs** - represent the file that contains the view model class for the `HomePage`
- **AboutViewModel.cs**- represent the file that contains the view model class for the `AboutPage`
- **ViewModelLocator.cs** - represent the file that class that helps to binding the `ViewModel` with the Page's `DataContext`

The resulting project structure is:



Project structure.

Home Page.xaml

(C) Copyright Nokia 2014. All rights reserved.

The `HomePage` will be the first page showed when the application is launched. This page contains the `WebBrowser` control to show this article and has the button "About Page".

This XAML defines the user interface for the Home page. Note that because we are using MVVM pattern, all logical code will be inside the view model `HomeViewModel` that is binding to `DataContext`.

```
<phone:PhoneApplicationPage
    x:Class="NokiaDev.AboutPageSample.Views.HomePage"
    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:i="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
    xmlns:cimbalino="clr-
namespace:Cimbalino.Phone.Toolkit.Behaviors;assembly=Cimbalino.Phone.Toolkit"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    DataContext="{Binding HomeViewModel, Source={StaticResource Locator}}"
    mc:Ignorable="d"
    shell:SystemTray.IsVisible="True">

    <!--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 Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="Nokia Developer Wiki Samples" Style="{StaticResource
PhoneTextNormalStyle}"/>
            <TextBlock Text="Article" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
            <TextBlock Text="About page for Windows Phone applications"
                       FontSize="{StaticResource PhoneFontSizeSmall}"
                       Foreground="{StaticResource PhoneAccentBrush}"
                       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">
            <phone:WebBrowser
Source="http://www.developer.nokia.com/Community/Wiki/Advanced_About_Page_for_Windows_Phone_apps"
/>
        </Grid>
        <i:Interaction.Behaviors>
            <cimbalino:ApplicationBarBehavior>
                <cimbalino:ApplicationBarIconButton Command="{Binding AboutCommand,
Mode=OneTime}"
                                                    IconUri="/Images/appbar.about.png"
                                                    Text="About" />
                <cimbalino:ApplicationBarBehavior.MenuItems>
                    <cimbalino:ApplicationBarMenuItem Command="{Binding AboutCommand,
Mode=OneTime}"
                                                      Text="About"/>
                </cimbalino:ApplicationBarBehavior.MenuItems>
            </cimbalino:ApplicationBarBehavior>
```


(C) Copyright Nokia 2014. All rights reserved.

```
        </i:Interaction.Behaviors>
    </Grid>
</phone:PhoneApplicationPage>
```

At the end of the XAML code, is possible to see the definition for the ApplicationBar that uses the ApplicationBarBehavior from Cimbalino Toolkit. The reason we did not used the Application Bar control from SDK is related with the fact that it is not a `FrameworkElement` and hence it cannot be used to apply a binding; this is a problem for anyone implementing the MVVM pattern.

See more about the `ApplicationBarBehavior` from Cimbalino Toolkit, here:

- Cimbalino Windows Phone Toolkit: ApplicationBarBehavior
- Cimbalino Windows Phone Toolkit: MultiApplicationBarBehavior

About Page.xaml

`AboutPage` represents the "About" or "Information" page itself. This page contains some controls for the fields described in the introduction and contains the buttons for the additional features.

The user interface for the About page is as follows. Note that because we are using MVVM pattern, all logical code will be inside the view model `AboutViewModel` that is binding to `DataContext`.

```
<phone:PhoneApplicationPage
    x:Class="NokiaDev.AboutPageSample.Views.AboutPage"
    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:i="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
    xmlns:cimbalino="clr-
namespace:Cimbalino.Phone.Toolkit.Behaviors;assembly=Cimbalino.Phone.Toolkit"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    DataContext="{Binding AboutViewModel, Source={StaticResource Locator}}"
    mc:Ignorable="d"
    shell:SystemTray.IsVisible="True">

    <!--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 Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="Nokia Developer Wiki Samples" Style="{StaticResource
PhoneTextNormalStyle}"/>
            <TextBlock Text="About" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
        </StackPanel>

        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel Margin="0,-12,0,24">
                <TextBlock Style="{StaticResource PhoneTextExtraLargeStyle}"
                           Text="About Page Sample"
                           Foreground="{StaticResource PhoneAccentBrush}" />
                <TextBlock Style="{StaticResource PhoneTextLargeStyle}">
                    by <Run Text="{Binding Author}" />
                </TextBlock>
```


(C) Copyright Nokia 2014. All rights reserved.

```
                <StackPanel Orientation="Horizontal" Margin="0,18,0,0">
                    <TextBlock FontWeight="Bold"
                               FontSize="{StaticResource PhoneFontSizeMedium}"
                               Style="{StaticResource PhoneTextNormalStyle}"
Text="Version:" />
                    <TextBlock Margin="0"  Text="{Binding Version}"
                               Style="{StaticResource PhoneTextNormalStyle}"
x:Name="AppVersion" />
                </StackPanel>

                <TextBlock Style="{StaticResource PhoneTextNormalStyle}"
                           FontSize="{StaticResource PhoneFontSizeMedium}"
                           Margin="12,12,0,0" FontWeight="Bold"
                           Text="Copyright (c) 2013 Nokia Developer Wiki." />

                <TextBlock Style="{StaticResource PhoneTextNormalStyle}"
                           Margin="12,24,0,0"  FontSize="{StaticResource
PhoneFontSizeSmall}"
                           TextWrapping="Wrap">
                     This application show an 'About page' sample for Windows Phone
applications.
                    <LineBreak/>See more about it, please see the following reference:
                </TextBlock>
                <HyperlinkButton
NavigateUri="http://www.developer.nokia.com/Community/Wiki/About_page_for_Windows_Phone_applications"
                                 Foreground="{StaticResource PhoneAccentBrush}"
                                 TargetName="_new" FontSize="{StaticResource
PhoneFontSizeSmall}"
                                 HorizontalAlignment="Left"
                                 Content="About page for Windows Phone applications" />

                <TextBlock Style="{StaticResource PhoneTextNormalStyle}"
                           Margin="12,24,0,12" FontWeight="Bold"
                           Text="More informations:"
                           FontSize="{StaticResource PhoneFontSizeMedium}" />

                <HyperlinkButton NavigateUri="http://www.developer.nokia.com/"
                                 Foreground="{StaticResource PhoneAccentBrush}"
                                 TargetName="_new"
                                 HorizontalAlignment="Left"
                                 Content="Website" />

                <HyperlinkButton NavigateUri="http://www.developer.nokia.com/"
                                 Foreground="{StaticResource PhoneAccentBrush}"
                                 TargetName="_new"
                                 HorizontalAlignment="Left"
                                 Content="Support" />

                <HyperlinkButton NavigateUri="http://www.developer.nokia.com/"
                                 Foreground="{StaticResource PhoneAccentBrush}"
                                 TargetName="_new"
                                 HorizontalAlignment="Left"
                                 Content="Privacy Policy" />

            </StackPanel>
        </Grid>

        <i:Interaction.Behaviors>
            <cimbalino:ApplicationBarBehavior>
                <cimbalino:ApplicationBarIconButton Command="{Binding RateCommand,
Mode=OneTime}"
                                                    IconUri="/Images/appbar.rate.png"
```


(C) Copyright Nokia 2014. All rights reserved.

```
                                                    Text="Rate it" />
                <cimbalino:ApplicationBarIconButton Command="{Binding
SendFeedbackCommand, Mode=OneTime}"

IconUri="/Images/appbar.reply.email.png"
                                                    Text="Feedback" />
                <cimbalino:ApplicationBarIconButton Command="{Binding
ShareToMailCommand, Mode=OneTime}"
                                                    IconUri="/Images/appbar.email.png"
                                                    Text="Email" />
                <cimbalino:ApplicationBarIconButton Command="{Binding
ShareSocialNetworkCommand, Mode=OneTime}"
                                                    IconUri="/Images/appbar.share.png"
                                                    Text="Share it" />
            </cimbalino:ApplicationBarBehavior>
        </i:Interaction.Behaviors>
    </Grid>
</phone:PhoneApplicationPage>
```

As in `HomePage`, we use the ApplicationBarBehavior from Cimbalino Toolkit to define the application bar with binding support.

HomeViewModel.cs

The Home Page view model is as below:

```
namespace NokiaDev.AboutPageSample.ViewModels
{
    using System;
    using System.Windows.Input;

    using Cimbalino.Phone.Toolkit.Services;

    using GalaSoft.MvvmLight.Command;

    /// <summary>
    /// The home view model.
    /// </summary>
    public class HomeViewModel
    {
        /// <summary>
        /// The navigation service.
        /// </summary>
        private readonly INavigationService _navigationService;

        /// <summary>
        /// Initializes a new instance of the <see cref="HomeViewModel"/> class.
        /// </summary>
        /// <param name="navigationService">
        /// The navigation service.
        /// </param>
        public HomeViewModel(INavigationService navigationService)
        {
            this._navigationService = navigationService;
            this.AboutCommand = new RelayCommand(this.ShowAbout);
        }

        /// <summary>
        /// Gets the about command.
        /// </summary>
        /// <value>
        /// The about command.
        /// </value>
        public ICommand AboutCommand { get; private set; }
```

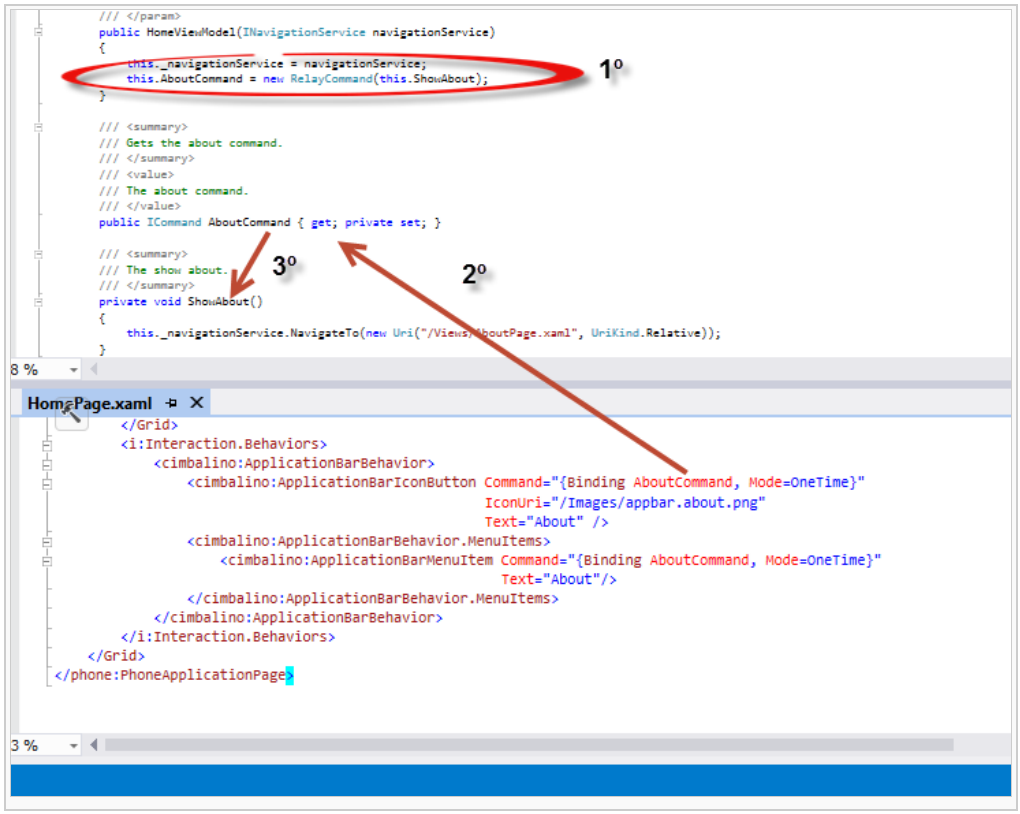

(C) Copyright Nokia 2014. All rights reserved.

```
        /// <summary>
        /// The show about.
        /// </summary>
        private void ShowAbout()
        {
            this._navigationService.NavigateTo(new Uri("/Views/AboutPage.xaml",
UriKind.Relative));
        }
    }
}
```

Here we used:

- `INavigationService` interface from Cimbalino toolkit (source code is here). This represents an interface for a service capable of handling the application navigation and the implementation for this interface will be NavigationService.
- `RelayCommand` class from MVVM Light toolkit (source code here).

A simple image that shows the flow:



AboutViewModel.cs

The About Page view model is as below:

```
namespace NokiaDev.AboutPageSample.ViewModels
{
    using System;
    using System.Windows.Input;

    using Cimbalino.Phone.Toolkit.Helpers;
    using Cimbalino.Phone.Toolkit.Services;

    using GalaSoft.MvvmLight.Command;

    /// <summary>
    /// The about view model.
    /// </summary>
    public class AboutViewModel
    {
        /// <summary>
        /// The email compose service.
```


(C) Copyright Nokia 2014. All rights reserved.

```
        /// </summary>
        private readonly IEmailComposeService _emailComposeService;

        /// <summary>
        /// The marketplace review service.
        /// </summary>
        private readonly IMarketplaceReviewService _marketplaceReviewService;

        /// <summary>
        /// The share link service.
        /// </summary>
        private readonly IShareLinkService _shareLinkService;

        /// <summary>
        /// The public application url.
        /// </summary>
        private readonly string _appUrl;

        /// <summary>
        /// The application manifest.
        /// </summary>
        private readonly ApplicationManifest _applicationManifest;

        /// <summary>
        /// Initializes a new instance of the <see cref="AboutViewModel"/> class.
        /// </summary>
        /// <param name="emailComposeService">
        /// The email Compose Service.
        /// </param>
        /// <param name="applicationManifestService">
        /// The application Manifest Service.
        /// </param>
        /// <param name="marketplaceReviewService">
        /// The marketplace review service
        /// </param>
        /// <param name="shareLinkService">
        /// The share Link Service.
        /// </param>
        public AboutViewModel(
            IEmailComposeService emailComposeService,
            IApplicationManifestService applicationManifestService,
            IMarketplaceReviewService marketplaceReviewService,
            IShareLinkService shareLinkService)
        {
            _emailComposeService = emailComposeService;
            _marketplaceReviewService = marketplaceReviewService;
            _shareLinkService = shareLinkService;
            RateCommand = new RelayCommand(this.Rate);
            SendFeedbackCommand = new RelayCommand(this.SendFeedback);
            ShareToMailCommand = new RelayCommand(this.ShareToMail);
            ShareSocialNetworkCommand = new RelayCommand(this.ShareSocialNetwork);
            _applicationManifest = applicationManifestService.GetApplicationManifest();
            _appUrl = string.Concat("http://windowsphone.com/s?appid=",
_applicationManifest.App.ProductId);
        }

        /// <summary>
        /// Gets the author.
        /// </summary>
        public string Author
        {
            get
            {
                return _applicationManifest.App.Author;
```


(C) Copyright Nokia 2014. All rights reserved.

```
            }
        }

        /// <summary>
        /// Gets the version.
        /// </summary>
        public string Version
        {
            get
            {
                return _applicationManifest.App.Version;
            }
        }

        /// <summary>
        /// Gets the rate command.
        /// </summary>
        public ICommand RateCommand { get; private set; }

        /// <summary>
        /// Gets the send feedback command.
        /// </summary>
        public ICommand SendFeedbackCommand { get; private set; }

        /// <summary>
        /// Gets the share social network command.
        /// </summary>
        public ICommand ShareSocialNetworkCommand { get; private set; }

        /// <summary>
        /// Gets the share to e-mail command.
        /// </summary>
        public ICommand ShareToMailCommand { get; private set; }

        /// <summary>
        /// The rate.
        /// </summary>
        private void Rate()
        {
            _marketplaceReviewService.Show();
        }

        /// <summary>
        /// The send feedback.
        /// </summary>
        private void SendFeedback()
        {
            const string To = "saramgsilva@gmail.com";
            const string Subject = "My Feedback";
            var body = string.Format(
                "Application {0}\n Version: {1}",
                _applicationManifest.App.Title,
                _applicationManifest.App.Version);
            _emailComposeService.Show(To, Subject, body);
        }

        /// <summary>
        /// The share social network.
        /// </summary>
        private void ShareSocialNetwork()
        {
            const string Message = "This application is amazing, should try it! See in";
            _shareLinkService.Show(_applicationManifest.App.Title, Message, new
Uri(_appUrl, UriKind.Absolute));
```


(C) Copyright Nokia 2014. All rights reserved.

```
        }

        /// <summary>
        /// The share to mail.
        /// </summary>
        private void ShareToMail()
        {
            const string Subject = "Nokia Developer Wiki Samples - About page";
            var body = string.Concat("This application is amazing, you should try it!
See in", _appUrl);
            _emailComposeService.Show(Subject, body);
        }
    }
}
```

Here we used:

- `RelayCommand` class (MVVM Light toolkit - source code here)

  Represents "A command whose sole purpose is to relay its functionality to other objects by invoking delegates. The default return value for the CanExecute method is 'true'. This class does not allow you to accept command parameters in the Execute and CanExecute callback methods".

- `IEmailComposeService` interface (Cimbalino toolkit - source code here)

  Represents an interface for a service capable of sending e-mail messages. The implementation is EmailComposeService

- `IMarketplaceReviewService` interface (Cimbalino toolkit - source code here)

  Represents an interface for a service capable of showing the marketplace review screen for an application and the implementation for this interface will be MarketplaceReviewService.

- `IShareLinkService` interface ((Cimbalino toolkit - source code here)

  Represents an interface for a service capable of sharing links over the social networks configured on the device. The implementation is ShareLinkService.

- `IApplicationManifest` interface (Cimbalino toolkit - source code here)

  Represents an interface for a service capable of reading from the application manifest. The implementation is ApplicationManifestService

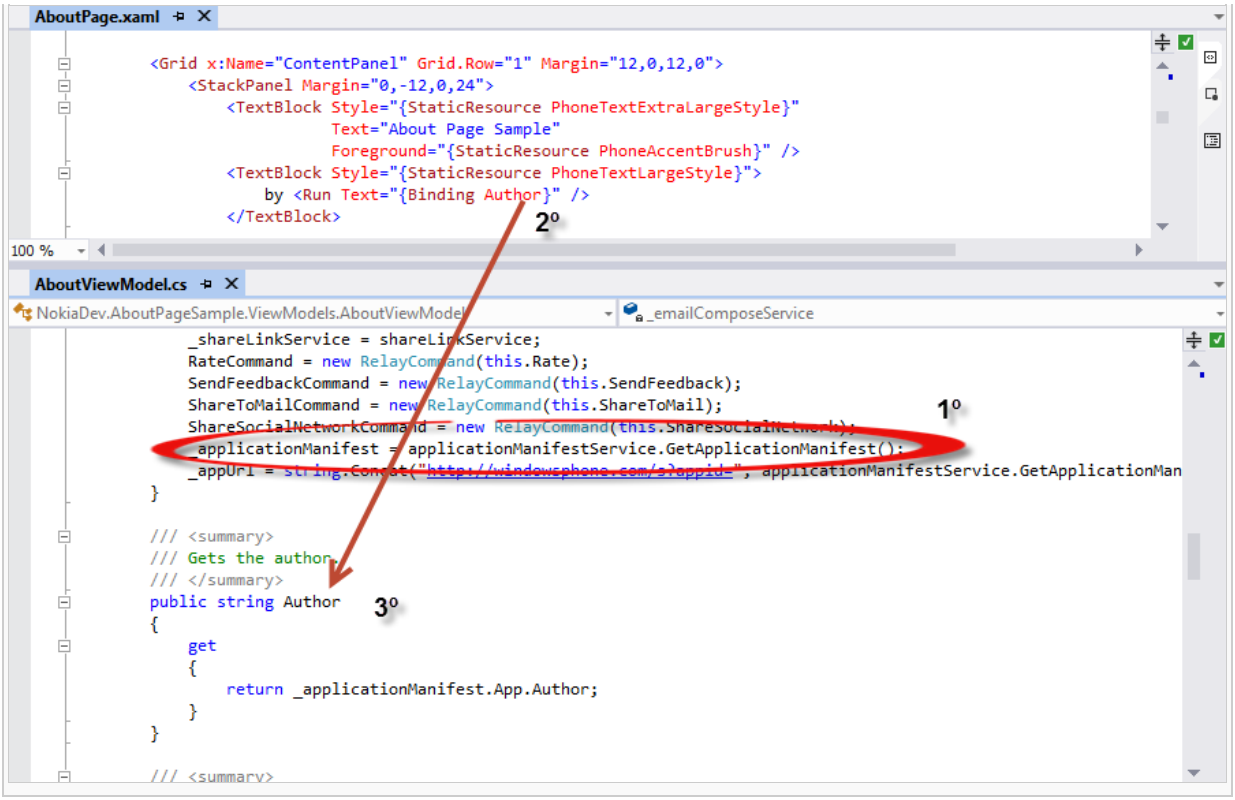
Each interface is injected in `ViewModel`, which are defined in `ViewModelLocator`. For example, for `NavigationService` we need to do:

```
SimpleIoc.Default.Register<INavigationService, NavigationService>();
```

This means when the `Ioc` container needs an implementation of `INavigationService` and will create an object of `NavigationService` (the idea is the same for the other dependencies).

As we saw in `HomeViewModel`, each application bar button will be binding to a command and it will be associated to an action. Let's see how it works for Author name:

(C) Copyright Nokia 2014. All rights reserved.



The `TextBlock`'s Text is binding to a property `Author` from `AboutViewModel` and this property will get the name of the author defined in Manifest file. There isn't a direct way to get it, but with `IApplicationManifest` from Cimbalino toolkit we don't need to worry about it, because this class get it for us without additional development.

The value for the `Version` is similar to the code for `Author`.

ViewModelLocator.cs

To help each page connect with its view model we use a `ViewModelLocator` to manage all view models and their dependencies.

```
namespace NokiaDev.AboutPageSample.ViewModels
{
    using Cimbalino.Phone.Toolkit.Services;

    using GalaSoft.MvvmLight.Ioc;

    using Microsoft.Practices.ServiceLocation;

    /// <summary>
    /// The view model locator.
    /// </summary>
    public class ViewModelLocator
    {
        /// <summary>
        /// Initializes a new instance of the <see cref="ViewModelLocator"/> class.
        /// </summary>
        public ViewModelLocator()
        {
             // starts the Ioc container
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            // register all dependecies
            if (!SimpleIoc.Default.IsRegistered<INavigationService>())
            {
                SimpleIoc.Default.Register<INavigationService, NavigationService>();
            }

            if (!SimpleIoc.Default.IsRegistered<IMarketplaceReviewService>())
            {
                SimpleIoc.Default.Register<IMarketplaceReviewService,
```


(C) Copyright Nokia 2014. All rights reserved.

```
MarketplaceReviewService>();
            }

            if (!SimpleIoc.Default.IsRegistered<IShareLinkService>())
            {
                SimpleIoc.Default.Register<IShareLinkService, ShareLinkService>();
            }

            if (!SimpleIoc.Default.IsRegistered<IApplicationManifestService>())
            {
                SimpleIoc.Default.Register<IApplicationManifestService,
ApplicationManifestService>();
            }

            if (!SimpleIoc.Default.IsRegistered<IEmailComposeService>())
            {
                SimpleIoc.Default.Register<IEmailComposeService, EmailComposeService>();
            }

            // register all view models
            SimpleIoc.Default.Register<HomeViewModel>();

            SimpleIoc.Default.Register<AboutViewModel>();
        }

        /// <summary>
        /// Gets the home view model.
        /// </summary>
        public HomeViewModel HomeViewModel
        {
            get
            {
                return ServiceLocator.Current.GetInstance<HomeViewModel>();
            }
        }

        /// <summary>
        /// Gets the about view model.
        /// </summary>
        public AboutViewModel AboutViewModel
        {
            get
            {
                return ServiceLocator.Current.GetInstance<AboutViewModel>();
            }
        }

        /// <summary>
        /// The cleanup.
        /// </summary>
        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

The constructor of the `ViewModelLocator` starts the `Ioc` container (`SimpleIoc`) and registers all dependencies and View Models.

This Ioc container manages all dependencies and injects them to each View Model. The `ViewModelLocator` contains one property for each View Model registered, because it will be used in binding with Page's `DataContext`.

Note: The reason we verify if each dependency is register doing:

(C) Copyright Nokia 2014. All rights reserved.

```
  if (!SimpleIoc.Default.IsRegistered<  ....  >())
            {...}
```

is because, in design time, without it the dependency will be register twice and throw and exception.

We add the `ViewModelLocator` to **App.xaml**.

App.xaml

The **App.xaml** and **App.xaml.cs** is the class created when the application starts.

In XAML, is possible to add global resources dictionaries or even simple object. For our sample we will add the `ViewModelLocator` object that will be a static field.

```
<Application
    x:Class="NokiaDev.AboutPageSample.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:viewModel="clr-namespace:NokiaDev.AboutPageSample.ViewModels">

    <!--Application Resources-->
    <Application.Resources>
        <viewModel:ViewModelLocator x:Key="Locator"  />
    </Application.Resources>

    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService
            Launching="ApplicationLaunching" Closing="ApplicationClosing"
            Activated="ApplicationActivated" Deactivated="ApplicationDeactivated"/>
    </Application.ApplicationLifetimeObjects>
</Application>
```

Benefits of MVVM

Using MVVM has many benefits:

- The view model implementation is independent of the services implementation like `ApplicationManifestService`, `NavigationService`, etc. Each time the view is changed only the XAML code needs to be changed. It is not necessary to change the behaviour just because the view changed (as long as the features are the same!);
- This approach allows us to implement unit test for the view model using fake services;
- Maintenance is much easier
- Separation of concerns and features allows cleaner/better code;

Because each toolkit is used by a lot of people it more likely that any issues will be fixed. It allows rapid and effective development.

Reusing an "About Page"

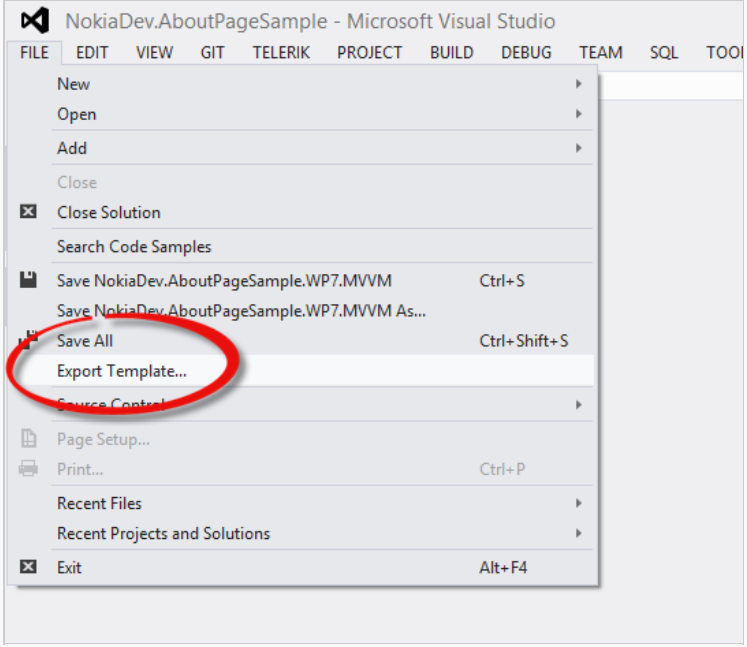
Almost every application we create will have an *About Page*, and these will be virtually the same. To simplify our development process, we might reuse this work by:

1. Creating item templates for: **AboutPage.xaml** and for **AboutViewModel.cs**
2. Creating project template for the project, allowing you to choose one of the targets.

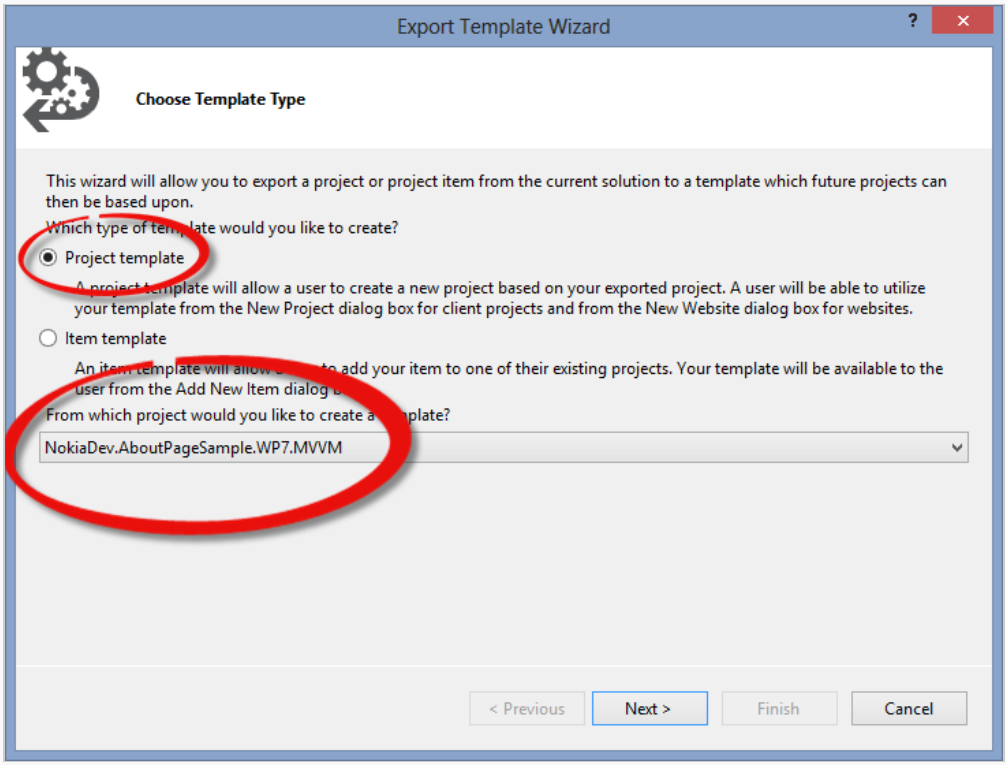
Because our project has dependencies from external references (Cimbalino Windows Phone Toolkit, MVVM Light Toolkit (PCL version)) and from the images contained in images folder, the second method (creating the project template) is a better approach. This is the option discussed below.

1. **Export the template project**

(C) Copyright Nokia 2014. All rights reserved.

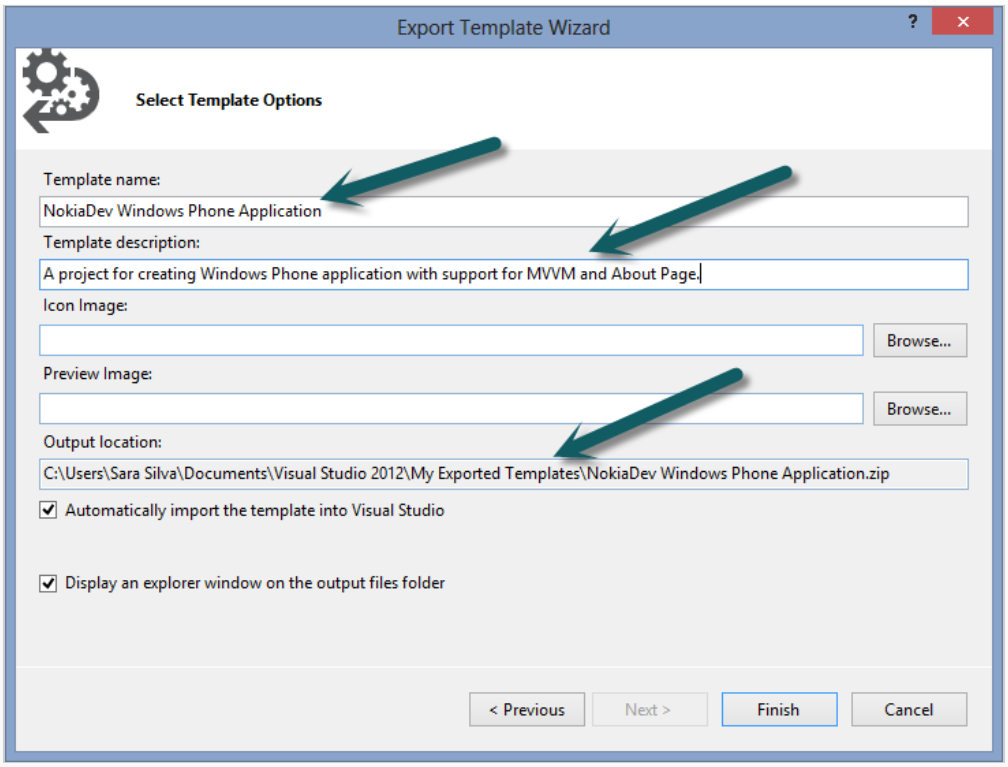


Selecting the "Export Template" option.

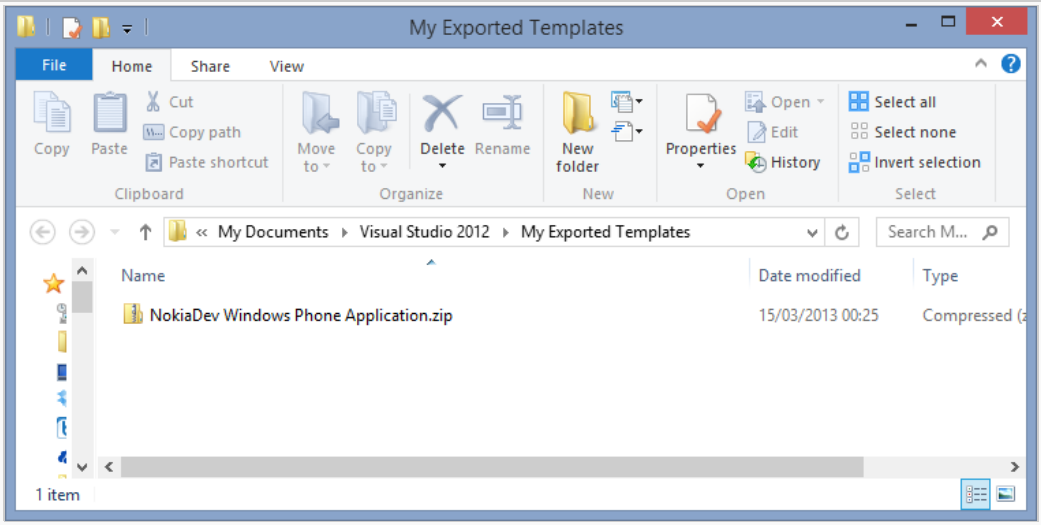


"Export Template Wizard" - Selecting the Project Template for the "NokiaDev.AboutPageSample.WP7.MVVM" project

(C) Copyright Nokia 2014. All rights reserved.



Defining the details for the new Template.



The *.zip resulting the "Export Template Wizard"

2. **Editing the exported template**

   After unzip the result file, we can find some files, *.cs and *.xaml files are common for us, and we recognize these. But there is an important file, with the extension *.vstemplate* (normally remaned for MyTemplate.vstemplate), which contains the definition for the project when is created. Here is an initial version, with generic definition that will be defined when the project is created.

```
<VSTemplate Version="3.0.0"
xmlns="http://schemas.microsoft.com/developer/vstemplate/2005" Type="Project">
  <TemplateData>
    <Name>NokiaDev Windows Phone Application</Name>
    <Description>A project for creating Windows Phone application with support
for MVVM and About Page.</Description>
    <ProjectType>CSharp</ProjectType>
    <ProjectSubType>
    </ProjectSubType>
    <SortOrder>1000</SortOrder>
    <CreateNewFolder>true</CreateNewFolder>
    <DefaultName>NokiaDev Windows Phone Application</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
    <LocationField>Enabled</LocationField>
    <EnableLocationBrowseButton>true</EnableLocationBrowseButton>
```


(C) Copyright Nokia 2014. All rights reserved.

```
    <Icon>__TemplateIcon.ico</Icon>
  </TemplateData>
  <TemplateContent>
    <Project TargetFileName="NokiaDev.AboutPageSample.csproj"
File="NokiaDev.AboutPageSample.csproj" ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true"
TargetFileName="App.xaml">App.xaml</ProjectItem>
      <ProjectItem ReplaceParameters="true"
TargetFileName="App.xaml.cs">App.xaml.cs</ProjectItem>
      <ProjectItem ReplaceParameters="false"
TargetFileName="ApplicationIcon.png">ApplicationIcon.png</ProjectItem>
      <ProjectItem ReplaceParameters="false"
TargetFileName="Background.png">Background.png</ProjectItem>
      <Folder Name="Images" TargetFolderName="Images">
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.about.png">appbar.about.png</ProjectItem>
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.email.png">appbar.email.png</ProjectItem>
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.rate.png">appbar.rate.png</ProjectItem>
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.reply.email.png">appbar.reply.email.png</ProjectItem>
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.share.png">appbar.share.png</ProjectItem>
        <ProjectItem ReplaceParameters="false"
TargetFileName="appbar.share.settings.png">appbar.share.settings.png</ProjectItem>
      </Folder>
      <ProjectItem ReplaceParameters="true"
TargetFileName="packages.config">packages.config</ProjectItem>
      <Folder Name="Properties" TargetFolderName="Properties">
        <ProjectItem ReplaceParameters="true"
TargetFileName="AppManifest.xml">AppManifest.xml</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="AssemblyInfo.cs">AssemblyInfo.cs</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="WMAppManifest.xml">WMAppManifest.xml</ProjectItem>
      </Folder>
      <ProjectItem ReplaceParameters="true" TargetFileName="Read-Me.txt">Read-
Me.txt</ProjectItem>
      <ProjectItem ReplaceParameters="false"
TargetFileName="Settings.StyleCop">Settings.StyleCop</ProjectItem>
      <ProjectItem ReplaceParameters="false"
TargetFileName="SplashScreenImage.jpg">SplashScreenImage.jpg</ProjectItem>
      <Folder Name="ViewModels" TargetFolderName="ViewModels">
        <ProjectItem ReplaceParameters="true"
TargetFileName="AboutViewModel.cs">AboutViewModel.cs</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="HomeViewModel.cs">HomeViewModel.cs</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="ViewModelLocator.cs">ViewModelLocator.cs</ProjectItem>
      </Folder>
      <Folder Name="Views" TargetFolderName="Views">
        <ProjectItem ReplaceParameters="true"
TargetFileName="AboutPage.xaml">AboutPage.xaml</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="AboutPage.xaml.cs">AboutPage.xaml.cs</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="HomePage.xaml">HomePage.xaml</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="HomePage.xaml.cs">HomePage.xaml.cs</ProjectItem>
      </Folder>
    </Project>
  </TemplateContent>
 </VSTemplate>
```


(C) Copyright Nokia 2014. All rights reserved.

Note: **TemplateData** contains all template definitions and **TemplateContent** contains the folders and files that will be added to the project when this is created.

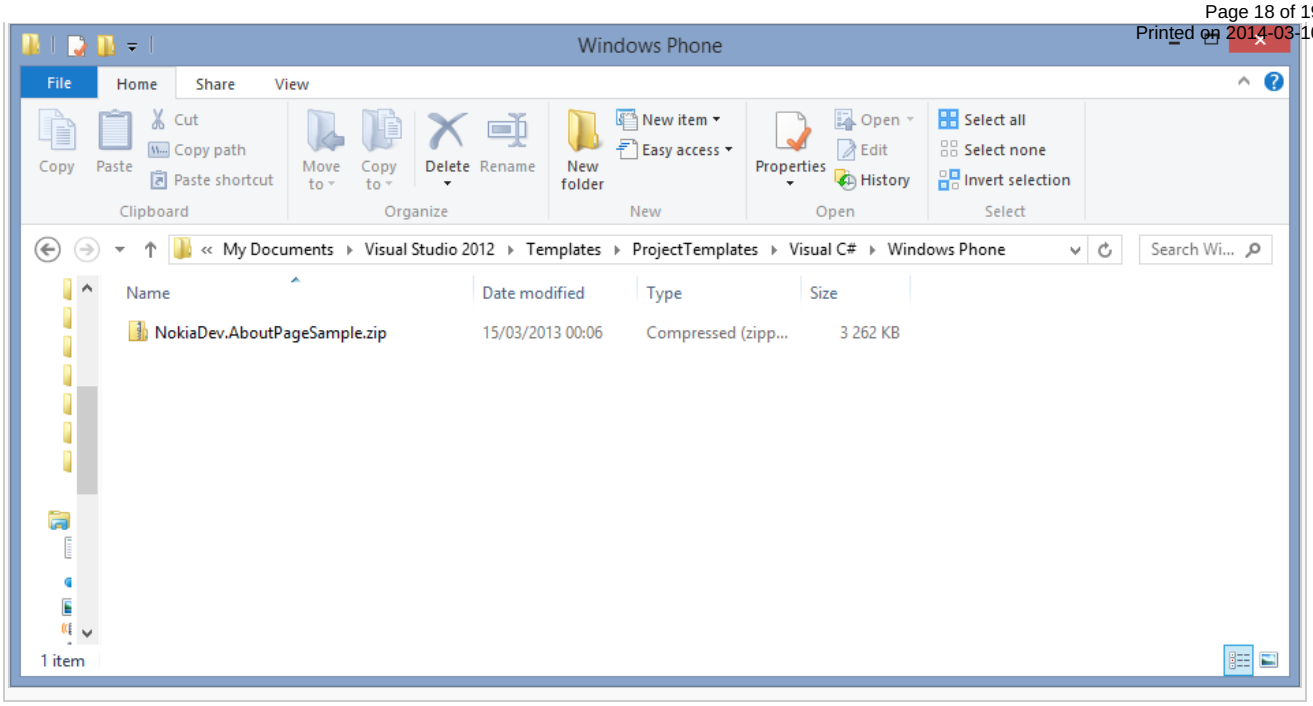
Name	Type	Compressed size	Password ...	Size	Ratio
Images	File folder				
Properties	File folder				
ViewModels	File folder				
Views	File folder				
__TemplateIcon.ico	ICO File	1 KB	No	10 KB	93%
App.xaml	Windows Markup File	1 KB	No	1 KB	60%
App.xaml.cs	CS File	2 KB	No	8 KB	79%
ApplicationIcon.png	PNG File	2 KB	No	2 KB	0%
Background.png	PNG File	4 KB	No	4 KB	5%
Cimbalino.Phone.Toolkit.2.1.nupkg	NUPKG File	301 KB	No	305 KB	2%
Microsoft.Bcl.1.0.16-rc.nupkg	NUPKG File	690 KB	No	755 KB	9%
MyTemplate.vstemplate	Visual Studio Project/Item...	2 KB	No	5 KB	78%
NokiaDev.AboutPageSample.csproj	Visual C# Project file	2 KB	No	8 KB	74%
packages.config	CONFIG File	1 KB	No	1 KB	49%
Portable.CommonServiceLocator.1...	NUPKG File	41 KB	No	45 KB	11%
Portable.MvvmLightLibs.4.1.27.1.n...	NUPKG File	48 KB	No	171 KB	73%
Read-Me.txt	Text Document	1 KB	No	1 KB	36%
Settings.StyleCop	STYLECOP File	1 KB	No	2 KB	78%
SplashScreenImage.jpg	JPG File	4 KB	No	10 KB	66%
Unofficial.Blend.Interactivity.1.0.0....	NUPKG File	2 154 KB	No	2 183 KB	2%

Tip: We added the *nupkg* (*Cimbalino.Phone.Toolkit.2.1.nupkg*, *Portable.MvvmLightLibs.4.1.27.1.nupkg*, *Unofficial.Blend.Interactivity.1.0.0.nupkg* and *Portable.CommonServiceLocator.1.2.2.nupkg*) that will be used for install the NuGet packages used in project. And is need to add theses reference in *.vstemplate file, at the bottom.

```
<VSTemplate Version="3.0.0"
xmlns="http://schemas.microsoft.com/developer/vstemplate/2005" Type="Project">
.....
<WizardExtension>
    <Assembly>NuGet.VisualStudio.Interop, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a</Assembly>
    <FullClassName>NuGet.VisualStudio.TemplateWizard</FullClassName>
  </WizardExtension>
  <WizardData>
   <packages repository="template">
    <package id="Cimbalino.Phone.Toolkit" version="2.1" targetFramework="wp71"
/>
    <package id="Portable.CommonServiceLocator" version="1.2.2"
targetFramework="wp71" />
    <package id="Portable.MvvmLightLibs" version="4.1.27.1"
targetFramework="wp71" />
    <package id="Unofficial.Blend.Interactivity" version="1.0.0"
targetFramework="wp71" />
 </packages>
  </WizardData>
</VSTemplate>
```

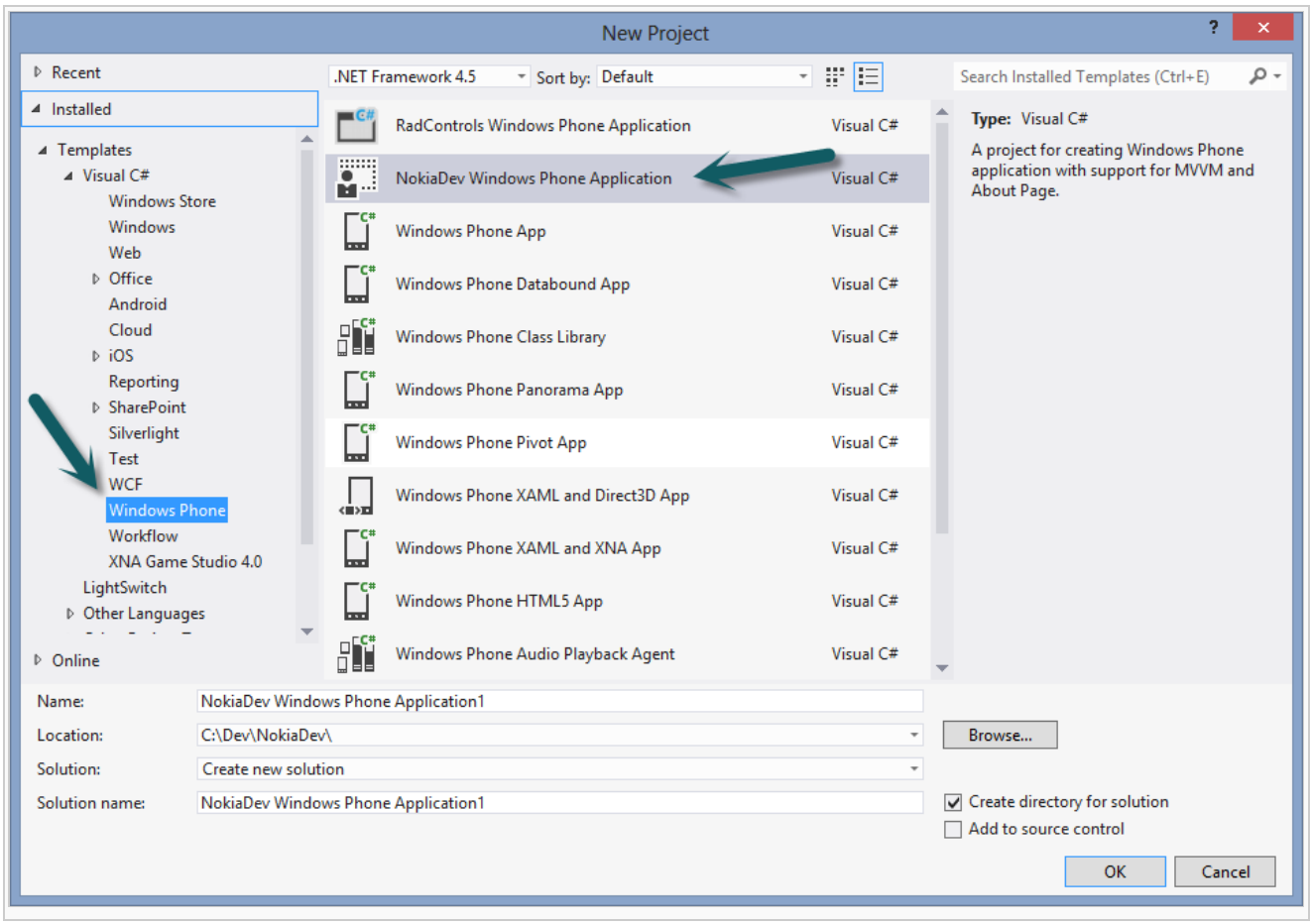
3. **Add the new template, for install, near all Windows Phone Templates (must be inside the folder with Windows Phone name)**

(C) Copyright Nokia 2014. All rights reserved.



Tip: The *.vstemplate file must be inside the root folder of the *.zip file, if not the template will not happer in list of templates.

4. **Install the new template**



Here is the result template: NokiaDev - Windows Phone Template

This template is created for Windows Phone SO 7.1 and can be upgraded for Windows Phone SO 8.0. The best approach, but outside the goal for the article, is to create an implementation of IWizard for selecting the correct target when the project is created.

Tip: Item and Project templates can be found in the following folders. Note that changes in these folders can lead to problems when running Visual Studio.

- *Item Templates* can be found in "C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\ItemTemplates\"

(C) Copyright Nokia 2014. All rights reserved.

- *Project Template* can be found in "C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\ProjectTemplates\"

Your ***Item Template*** or ***Project Template*** should be added in C:\Users\{USERNAME}\Documents\Visual Studio 2012\Templates

References:

- (MSDN) Creating Project and Item Templates
- (MSDN) Templates for Creating Templates
- (MSDN) Packages in Visual Studio Templates
- (MSDN) How to: Use Wizards with Project Templates

Conclusion

In conclusion, an about page makes the user informed about all useful details about the application.This allow to add additional features like rate, feedback and share which increases the quality and can be great for marketing. User opinion is the best feedback we can get for have a great application, for this reason is recommended to have one. To ensure a quick development in repeated task is recommended to reuse code.

Additional references

The following articles are old and to some extent out of date. However the fundamental concepts remain sound:

- Building a detailed About page for your Windows Phone application
- A Standard About Page for Windows Phone 7 applications

(C) Copyright Nokia 2014. All rights reserved.