Put a professional polish on your Windows Phone 7 applications

# Windows Phone 7 Recipes

A Problem-Solution Approach

Fabio Claudio Ferracchiati | Emanuele Garofalo

Apress®

# Windows Phone 7 Recipes

A Problem-Solution Approach

Fabio Claudio Ferracchiati
Emanuele Garofalo

Apress®

**Windows Phone 7 Recipes**

Copyright © 2011 by Fabio Claudio Ferracchiati and Emanuele Garofalo

ISBN 978-1-4302-3371-8

ISBN 978-1-4302-3372-5 (eBook)

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com` or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at `www.apress.com/bulk-sales`.

The source code for this book is available to readers at `www.apress.com`. You will need to answer questions pertaining to this book in order to successfully download the code.

*To Simona and Claudia, women of mine  –Fabio*

*To my mommy and daddy, my lighthouse in the fog – Emanuele*

# Contents at a Glance

# Contents

# About the Authors

■**Fabio Claudio Ferracchiati** is a prolific writer and technical reviewer on cutting-edge technologies. He has contributed to many books on .NET, C#, Visual Basic, SQL Server, Silverlight, and ASP.NET. He is a .NET Microsoft Certified Solution Developer (MCSD) and lives in Rome, Italy. He is employed by Brain Force.

■**Emanuele Garofalo** was born at Torre del Greco (Naples), Italy and now lives in Rome. He is an active member of the DotNetCampania community, and works with Windows Presentation Foundation (WPF), Silverlight, Windows Communication Foundation (WCF), and Language Integrated Query (LINQ) as principal technologies.

# About the Technical Reviewer

■**Simona Nasetti** is an expert Microsoft Dynamics CRM and Microsoft SQL Server Business Intelligence consultant. She graduated with a mathematics degree and works at Agic Technology (`www.agic.it`) in Rome, Italy, where she creates vertical solutions and reports for the company's clients.

# Acknowledgments

First of all, I'd like to thank Ewan Buckingham. During these years we had working experinces together and he has been always patient, kind, and helpful.  I'd also like to thank Emanuele for the great job he did working with me to complete this book. Moreover, the Apress guys, each of you have been great. So, thanks to Jessica Belanger, Dominic Shakeshaft, Christine Ricketts, and Sharon Wilkey. Finally, my wife Simona for her efforts especially trying to quiet my daughter's screams while I was writing the book. Wait, thanks to V's authors, they distract me with very cool episodes.

Fabio Claudio Ferracchiati

Thanks to Alessandra for her patience during the writing of this book, to Fabio for the opportunity to write this book, to Ewan, Jessica, Sharon and all the staff of Apress for their support, and to Michele Aponte and all  the DotNetCampania community for the motivation to study new technologies with the spirit of knowledge-sharing.

Emanuele Garofalo

# About This Book

Are you interested in Windows Phone 7 configuration and development? Learn to build, configure, and distribute your applications through a wide variety of step-by-step recipes. This book contains extensive code samples and detailed walk-throughs.

CHAPTER 1

# Introduction to Windows Phone 7 Application Development

This chapter introduces Windows Phone 7, including its device hardware characteristics and software development tools. After this introduction, you will learn how to create simple applications and how to deploy them. Finally, you are going to look at the application's distribution via Windows Phone Marketplace. The recipes in this chapter describe how to do the following:

- 1-1: Examine Windows Phone 7 hardware
- 1-2: Examine Windows Phone 7 development tools
- 1-3 and 1-4: Create a simple Windows Phone 7 Silverlight and XNA application
- 1-5 and 1-6: Deploy a Windows Phone 7 application on both the emulator and device
- 1-7: Put a Windows Phone 7 application into Windows Phone Marketplace
- 1-8: Create a Windows Phone 7 trial application

## 1-1. Examining Windows Phone 7

### Problem

You have just bought your new Windows Phone 7 device and would like to start developing applications. You need to know the device's hardware characteristics such as screen resolution and memory capability—but also which sensors and services it provides. Finally, you need to understand what the Windows Phone 7 operating system provides in order to integrate your application in the best way.

### Solution

If you have Windows Mobile development experience, please erase it! Joking aside, Microsoft has provided a brand new operating system for its new smartphone: Windows Phone 7. This operating system has been written from scratch in order to reach—and sometime go beyond—other operating systems' functionalities.

To make an operating system that is reliable and fast and has high performance, Microsoft has dictated hardware requirements. So, every Windows Phone 7–compatible phone on the market right now and in the future has (or will have) at least minimum common hardware characteristics. For us as developers, this is great news, because we can write code having some certainty of what the smartphone provides.

The Windows Phone 7 device provides a screen resolution of 480×800 pixels in portrait orientation. In the future, mobile vendors plan to release smartphones with smaller screens having a resolution of 320×480 pixels. Having this in mind, you can create a game and draw your sprites knowing that your game will be played on a screen with that resolution—so no scale operations, screen text adaptation, and so forth will be necessary. But even for classic applications showing, for example, text boxes and buttons, this resolution is useful for drawing rich user interfaces.

Every phone provides three hardware buttons, usually at the bottom of the screen, as shown in Figure 1-1. They are the Back button, the Start button, and the Search button. The leftmost button is used to go back to the previous application (just like the Back button on an Internet browser). The middle button is used to close the current application and to show the Start menu so that the user can open another application. The rightmost button is used to access the start screen and start a search (for example, a search into the phone content for contacts or a search on the Bing site).

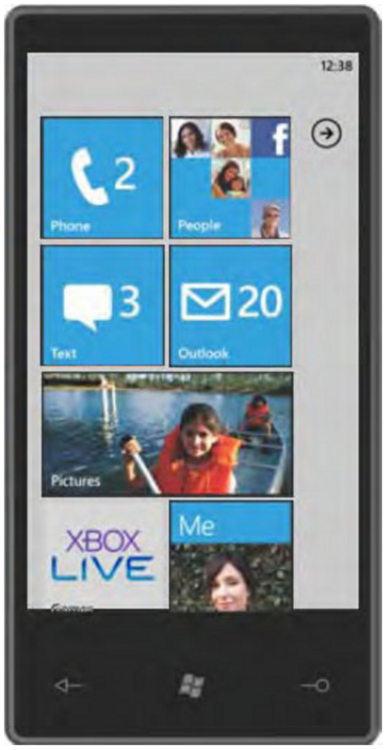


***Figure 1-1.*** *An image of a generic Windows Phone 7 device*

From a developer's point of view, it is important to understand the impact that these buttons have on an application. When each button is pressed, the running application is either deactivated or killed. A developer has to provide code that responds to those events, perhaps saving data in isolated storage (an application's disk-dedicated storage). To redisplay the application, perhaps after the user pushes the Back button, code has to have been written in order to re-create the same situation present before the deactivation. You can see more on this in Chapter 2.

Windows Phone 7 devices have a Soft Input Panel (SIP) that enables users to write text into text boxes. A hardware keyboard is optional. In both cases, the application will receive text input in the same manner. The same is true for key pressure events. The SIP is shown automatically by Windows Phone 7 when text input is required by the user.

In Figure 1-1, you can see the Windows Phone 7 starting page and its new *Metro* user interface. Microsoft designers, with users' feedback, have preferred to put the accent on content and information instead of eye-catching graphics. So the screen is populated with something similar to either metro or airport banners. Every square and rectangle is called a *live tile* and gives access to the *hub*. Each live tile is updated in real time with information taken from the hub. The hub is a sort of aggregator to group similar information such as group photos taken from the web, from the phone itself, and from social networks. For example, the Office tile will show counter indicating the number of incoming e-mail when a new e-mail arrives. So the hub contains an aggregation of information that is both local (on the phone) and remote (on the cloud and from the Internet). For example, the *Pictures hub* contains photos taken from the internal camera and from social networks such as Facebook. There are six hubs provided with Windows Phone 7:

- People
- Office
- Pictures
- Music and Videos
- Marketplace
- Games

By the way, the phone is completely customizable, so you can remove live tiles, add your preferred ones, move tiles, and so on. Users can choose between two graphics themes: dark or light. Each presents a different background color (black and white, respectively), which is important to be aware of as you draw your icons, buttons, and images for an application.

The user can interact with Windows Phone 7 by using its multi-touch screen. Using your fingers to perform various gestures such as taps, you can move the tiles, zoom in and zoom out on text and pictures, and so on. Every vendor that produces Windows Phone 7 devices must provide at least a four-point multi-touch capacitive screen so that you can use at least four fingers on the touch screen.

The Windows Phone 7 device ships with 256 MB or more of RAM and with 8 GB or more of flash storage. The CPU is an ARMv7 with at least 1 GHz of frequency.

Finally, the Windows Phone 7 device provides sensors and services to bring the user experience to the next level. Here is a list of the most important ones:

*A-GPS*: This sensor is the Assisted Global Positioning System. It enables users to retrieve their position in the world in terms of longitude and latitude coordinates taken from both satellite services and cell-site triangulation. The latter is less accurate because it represents the nearest radio network from the phone position but it is useful when satellite signals are low or absent.

*Accelerometer*: This sensor enables programs to understand when the phone has been moved—for example, either because the user has taken it from the desk to respond to a call, or worse, the phone is falling from the user's hands!

*Wi-Fi*: This sensor enables the phone to connect to a Wi-Fi spot for an Internet connection.

*Camera*: This sensor enables users to take photos and videos through a 5-megapixel (or more) camera with flash.

*Office*: This service is not so advertised, but every phone has a very usable and powerful version of Microsoft Office with its common applications such as Word, Excel, Outlook, and PowerPoint.

*Location*: Thanks to this service, a user can be located, and that -user's position can be represented via Bing Maps.

*Push Notifications*: This is a great service that prevents phone to polling information from the Internet. The phone waits to receive notifications from programs that live outside the phone avoiding to continually going to search for new informations.

Developers can use all these sensors and services together to create innovative applications and sell them on Windows Phone Marketplace. They do not have to worry about hardware differences (for example, whether a certain cell model has the accelerometer) because every Windows Phone 7 has the same minimum features.

## 1-2. Understanding the Development Tools

### Problem

You want to start developing for Windows Phone 7. You want to know which tools and which languages you have to use to make an application.

### Solution

You have to download the Microsoft Windows Phone Developer Tools.

### How It Works

We started Recipe 1-1 saying that if you have Windows Mobile development experience, it is better to erase it! This is a joke, of course, but it is not completely false. In Windows Phone 7 development, you don't have the freedom to create low-level applications with C or C++ languages. Using .NET is the only way allowed by Microsoft to develop your applications for Windows Phone 7. Even if you find a way to go around this limitation—let's say by injecting some Intermediate Language (IL) code at runtime—you still have to remember that every application will be distributed by Windows Phone Marketplace. And, of course, before users can find your application on Marketplace, that application has to go through different approval steps, and you can be sure that any non-.NET application would not pass the certification process.

You can create two kinds of applications: Silverlight for Windows Phone and XNA for Windows Phone. The former uses a custom Silverlight 3 version in which Microsoft has added some specific features. The latter uses XNA libraries and is targeted at creating videogames. You can combine both technologies in your application, with the only limitation being the user interface; you can't draw controls by using Silverlight and use XNA to draw sprites at the same time. On the other hand, you can use Silverlight for the user interface and XNA libraries to provide full access to media storage on the phone, to capture audio, and more.

C# is actually the only language that has full support on both Silverlight and XNA technologies. At the time of this writing, with Visual Basic, you can develop only Silverlight applications.

To start developing, you first have to download the Windows Phone Developer Tools from `http://go.microsoft.com/fwlink/?LinkID=189554`. This setup includes Visual Studio 2010 Express for Windows Phone, Windows Phone 7 Emulator, Silverlight Tools, XNA 4, and Microsoft Expression Blendfor Windows Phone. If you already have Visual Studio 2010 installed on your machine, the setup will install only the necessary files and you will see new project templates the next time you start the development tool.

---

■ **Note** At the time of this writing, Visual Studio 2010 Express doesn't support Visual Basic. You must have Visual Studio 2010 Professional or Superior to use Visual Basic. You can always download a Visual Studio 2010 trial version.

---

Let's see the necessary steps to install the Microsoft Windows Phone Developer Tools:

1. Launch the installer (`vm_web.exe`) after having downloaded it.
2. Accept the license agreement.
3. Optionally, choose the Customized installation so you can select a folder in which to install the tools.
4. Wait for the installer to download all the necessary files from the Internet. The number of files downloaded depends on what the installer finds already in your operating system.
5. If you have to install the developer tools on machines not connected to the Internet, you can use the ISO version from `http://go.microsoft.com/fwlink/?LinkId=201927`.

The next step is to download the Windows Phone Developer Tools October 2010 update, which includes some updates such as the Windows Phone Connection Tool, some changes to Bing Maps for the Windows Phone Silverlight control, and a tool to detect phone capabilities. By the way, since things are moving so fast in the Windows Phone panorama, Microsoft's App Hub site at `http://create.msdn.com/en-US` should be the starting point for every developer.

## 1-3. Creating a Simple Silverlight Windows Phone 7 Application

### Problem

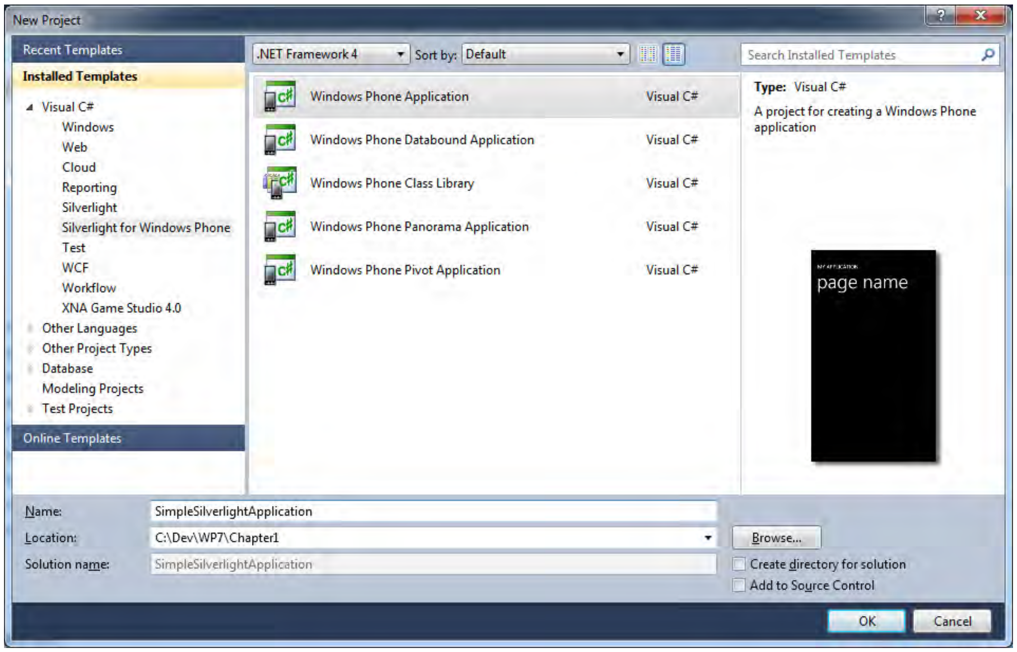
You have to create a Windows Phone 7 application by using Silverlight.

### Solution

Use Visual Studio 2010 (either the Express, Professional, or Superior edition). Use the Windows Phone Application project template.

### How It Works

After opening Visual Studio 2010, you have to create a new project. From the File menu, choose New Item ➤ Project item (or press Ctrl+Shift+N). Figure 1-2 shows the dialog box that appears after launching the New Project command.



.NET Framework 4 Sort by: Default

| | |
|---|---|
| Windows Phone Application | Visual C# |
| Windows Phone Databound Application | Visual C# |
| Windows Phone Class Library | Visual C# |
| Windows Phone Panorama Application | Visual C# |
| Windows Phone Pivot Application | Visual C# |

Type: Visual C#

A project for creating a Windows Phone application

Name: SimpleSilverlightApplication

Location: C:\Dev\WP7\Chapter1

Solution name: SimpleSilverlightApplication

Create directory for solution

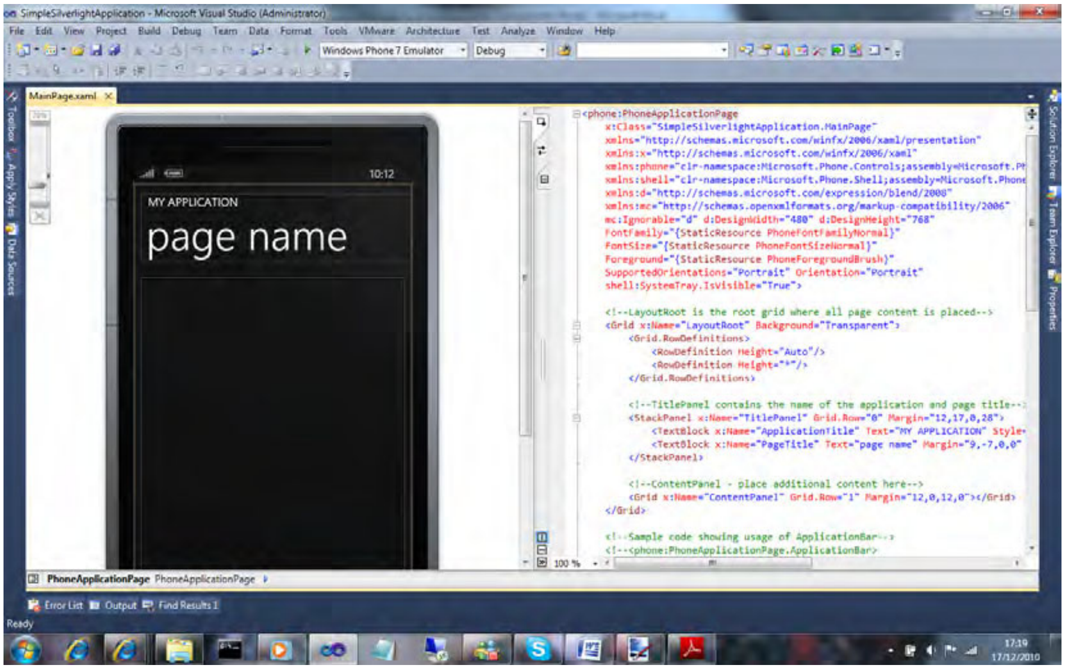
Add to Source Control

*Figure 1-2. Visual Studio 2010 New Project dialog box*

From the Installed Templates on the left, select Silverlight for Windows Phone. There are five project templates provided by Visual Studio 2010:

- *Windows Phone Application* creates a skeleton code for a generic phone application; no controls or other stuff are added.
- *Windows Phone Databound Application* creates a Windows Phone application, adding List and Navigation controls.
- *Windows Phone Class Library* creates a skeleton code for an external assembly specific to Windows Phone 7.
- *Windows Phone Panorama Application* creates an application including the Panorama control (see more on that in Chapter 3, Recipe 3-7).
- *Windows Phone Pivot Application* creates an application including the Pivot control (see more on that in Chapter 3, Recipe 3-7).

Select the Windows Phone Application project template and type `SimpleSilverlightApplication` in the project's Name text box. Choose a Location where to save the project and then click the OK button. Wait while Visual Studio 2010 writes every file and folder, and after a few seconds you should have `MainPage.xaml` opened in the integrated development environment (IDE) of Visual Studio 2010 (see Figure 1-3).


SimpleSilverlightApplication - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Data Format Tools VMware Architecture Test Analyze Window Help
Windows Phone 7 Emulator
Debug
MainPage.xaml
MY APPLICATION
page name
10:12
PhoneApplicationPage PhoneApplicationPage
Error List Output Find Results 1
Ready
Solution Explorer
Team Explorer
Properties
Toolbox
Apply Styles
Data Sources
100 %
17:19
17/12/2010

```
<phone:PhoneApplicationPage
    x:Class="SimpleSilverlightApplication.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.Ph
    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">

    <!--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=
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
        </StackPanel>

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

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
```

*Figure 1-3.* `MainPage.xaml` *ready to be edited*

## The Code

The project contains two main files: `App.xaml` and `MainPage.xaml`. Two classes are created: the App class and the MainPage class (see class diagram in Figure 1-4). The other files are resources such as a splash screen image, background image, and the application icon. Finally, there is an application manifest file called `WMAppManifest` that contains application data such as the application's title, the resource names, and so forth. It also includes a list of capabilities that you have to specify when you want to use a particular phone feature. For example, if you want to use the phone microphone in your application, you have to add the `ID_CAP_MICROPHONE` capability. The file comes with more than ten capabilities already defined in it; you should remove the ones you don't use.

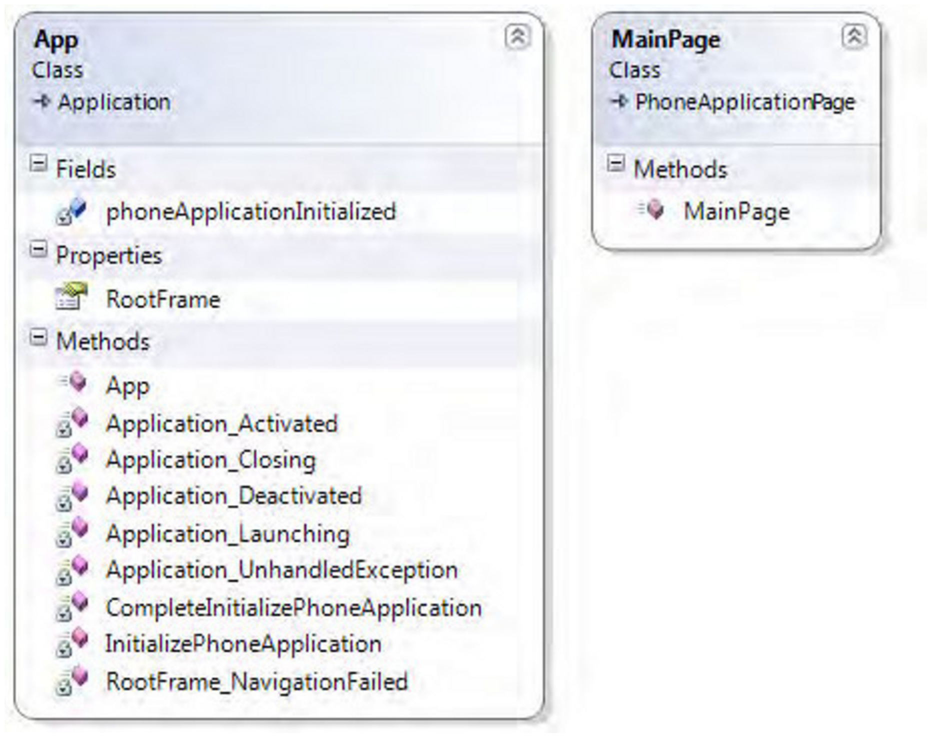


***Figure 1-4.*** *The class diagram for the App and MainPage classes.*

Let's focus our attention on the main two files. The `MainPage.xaml` file contains the Extensible Application Markup Language (XAML) markups that define the main page. At the beginning of the code, all the namespaces used by the application are declared.

```
<phone:PhoneApplicationPage
    x:Class="SimpleSilverlightApplication.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"

….
```

The root tag is PhoneApplicationPage, which is the class from which our application derives. The prefix phone: is necessary because the PhoneApplicationPage name is defined in the namespace clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone. The x:Class attribute states that the MainPage class is defined in the code-behind and is included in the SimpleSilverlightApplication namespace. The first namespace in the XAML code is the main Silverlight namespace; the x namespace contains definitions of all extra XAML elements not defined in the previous namespace. The shell, d, and mc namespaces are specific to the Windows Phone application and contain markups for shell instructions, Microsoft Expression Blend, and the Visual Studio designer.

The other attributes of the <phone:PhoneApplicationPage> markup are used to define the application's orientation, font, and colors. It is worth noting the use of static resources provided by the Windows Phone resource dictionary (see http://msdn.microsoft.com/en-us/library/ff769552(v=vs.92).aspx for the full list of available resources).

```
….
    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">
```

Then the code includes a grid with two rows. In the first row is a stack panel with two text blocks, and in the second row is a content panel where you can add your controls.

```
    <!--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>
```