

Windows Phone 7.5



Apps

In Record Time

TANZIM SAQIB

REVISION HISTORY

First made available online:
December 16, 2011

ABOUT THE AUTHOR

Tanzim Saqib is an industry veteran, multiple times Microsoft "Most Valuable Professional" award winner, long time .NET pirate and a renowned Software Architect. His programming experience dates back to 1995. He worked for outsourcing companies that developed software for major financial institutes in the US, built paperless University system, worked for couple world-renowned startups Pageflakes and Sitemasher, .NET product vendor, Telerik,

and finally designed architectures in one of the largest enterprises, British Telecom. He is also an open source activist, technology speaker, book reviewer and ACM online programming contest problemsetter. He can be reached:

Email: me@TanzimSaqib.com
Website: <http://TanzimSaqib.com>
Facebook: <http://fb.me/SaqibRocks>
Twitter: [TanzimSaqib](https://twitter.com/TanzimSaqib)



Windows Phone 7.5 Apps In Record Time | © Tanzim Saqib 2011. All rights reserved. Unauthorized distribution, redistribution, production, print, partial print of this work is strictly prohibited, and such acts will be considered breaching Copyright law, and shall be prosecuted.

Preface

Microsoft has come up with a brand new concept of phone. They have taken nothing from the predecessor or the competitors. This is a completely new start in terms of user interface and user experience. From its inception (7.0) to latest "Mango" release (7.5), it has never ceased to surprise with wealth of new innovative features and strong partnerships, which only indicates a very bright future ahead of this new mobile operating system. You are already familiar with the

glorifications and praises about Windows Phone. Otherwise, you wouldn't be interested in it. Let's cut the crap and start writing apps!

Oh—one last point, this micro eBook was written as briefly as possible to act as a jump starter for those who have intermediate level of practical experience in C#.

Tanzim Saqib | December 16, 2011

Platform and Tools

Microsoft's all development platforms and tools are based on .NET and Visual Studio respectively. Windows Phone's are no exception. Windows Phone tooling gives all that you need, including new Visual Studio project types, integrated Phone emulator to run, test, set breakpoints and debug your apps. You can build apps and games using: Silverlight, suitable for regular apps, and XNA Framework, which is suitable for high performance graphics rendering such as games. We will focus on apps only in here, so expect no XNA coverage. Navigate to <http://create.msdn.com> and grab the free tools. That's all you need!

Documentation

Now that you have tools, you will want to create your first project, and documentation of course: <http://bit.ly/dq0uqf>

First Project: Solution Structure

File -> New -> Project -> Silverlight for Windows Phone -> Windows Phone Application. After the solution has been prepared for you, first thing that should catch your eyes is the MainPage.xaml. This is equivalent of Default.aspx if you are from Web background, and Form1.cs if you are a Windows Forms guy or girl. So far so good. But, like Web form, it has a XML-like view (known as XAML) and a codebehind. Let's open the XAML file. XAML is an extensible version of XML format particularly designed for constructing WPF and Silverlight user interfaces. Learn XAML from here: <http://bit.ly/sNzLza>

Like regular XML schema references you will also find many in that XAML file. They are required for resources and controls are used in the page, much like control namespaces referenced in the WebForms. You will notice that there is a root element, like XML must have one, named Grid with x:Name="LayoutRoot." Yes, that's how you name an element (call them controls if you will). Obviously, you do have the same Toolbox, Properties, Solution Explorer and similar design experience as WebForms and Windows Forms.

Scroll a bit, and you will see three elements, two of which make sense to us: Two *TextBlocks* having *ApplicationTitle* and *PageTitle*. *TextBlocks* are *Labels*, only more powerful! Last thing we see is another *Grid* named *ContentPanel*.

Hello World

Now drag and drop a *Button* element into that *Grid*, and see how the XAML changes accordingly. Double click the *Button* and as you expect it will open the codebehind window for you to write an even handler for button click. We will start with a hello to the world:

```
MessageBox.Show("Hello World!");
```

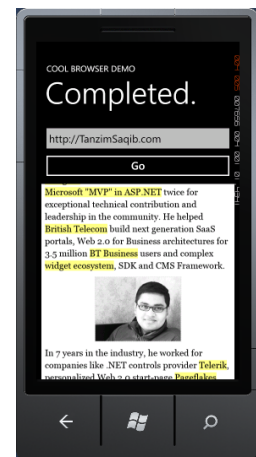
Let's go ahead and enhance this a bit. Drop a *TextBox*, *Button* and *WebBrowser* control with name *txtUrl*, *btnGo* and browser respectively. Add some code:

```
private void btnGo_Click(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(txtUrl.Text))
    {
        MessageBox.Show("Please type an URL first.");
        return;
    }

    PageTitle.Text = "Loading...";
    browser.Navigate(new Uri(txtUrl.Text));
}

private void browser_LoadCompleted(object sender, NavigationEventArgs e)
{
    PageTitle.Text = "Completed.";
}
```

Congratulations! You have built your first app.



Show your own HTML instead of a webpage:

```
browser.NavigateToString("<html><body><h1>Good morning! Where's my PC?</h1></body></html>");
```

Call JavaScript method from within your app:

```
browser.InvokeScript("my_cool_method", "parameter1", "parameter2", "parameterN");
```

Get notified by an event in the app, on a call from JavaScript code:

```
browser.ScriptNotify
```

Fun with Map

Let us have some fun with *Map* element, look at the following code and follow the comments in line.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    // set map to a longitude and latitude at zoom 10
    map1.SetView(new GeoCoordinate(23.723, 90.4086), 10);
    map1.Mode = new RoadMode(); // change to Road view
    map1.Mode = new AerialMode(); // to Ariel view
    ++map1.ZoomLevel;
}

private void map1_Hold(object sender, GestureEventArgs e)
{
    var point = e.GetPosition(map1); // get position on the screen
    GeoCoordinate location; // translate screen position to geo location
    map1.TryViewportPointToLocation(point, out location);
    MessageBox.Show(string.Format("Tapped Lat/Long: {0:0.000}/{1:0.000}", location.Latitude, location.Longitude));
}
```

GPS

Now that we have learned how to use elements from codebehind, let us see how we can access Windows Phone sensors, among which Global Positioning System (GPS) data is particularly interesting:

```
private readonly GeoCoordinateWatcher _GeoWatcher;

public MainPage()
{
    InitializeComponent();

    _GeoWatcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);
    _GeoWatcher.Start();
}

private void button1_Click(object sender, RoutedEventArgs e)
{
    var position = _GeoWatcher.Position;
    MessageBox.Show(string.Format("Current Lat/Long/Alt: {0:0.000}/{1:0.000}/{2:0.000}", position.Location.Latitude, position.Location.Longitude, position.Location.Altitude));
}
```

You need to use Additional tools available in the phone Emulator to enter some fake data and feed it to the app, so that you can simulate your app's behavior in actual condition. We will take a stab at it in the *Accelerometer* section. A point to remember while programming with sensors is that they use device power and can drain it pretty quickly.

Accelerometer

Let us access 3D position data of the phone. We need a reference to Microsoft.Devices.Sensors:

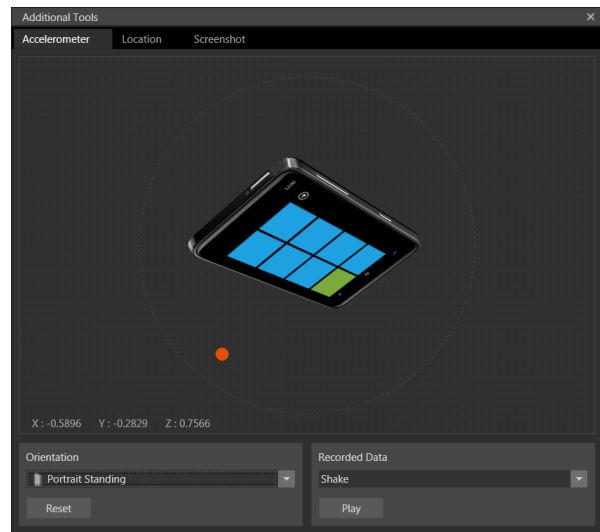
```
Accelerometer accel;

private void btnActivateAccel_Click(object sender, RoutedEventArgs e)
{
    accel = new Accelerometer();
    accel.CurrentValueChanged += accel_CurrentValueChanged;
    accel.Start();
}
```

```
void accel_CurrentValueChanged(object sender, SensorReadingEventArgs<AccelerometerReading> e)
{
    var reading = e.SensorReading.Acceleration;
    // access reading.X, reading.Y, reading.Z, e.SensorReading.Timestamp
}
```

Feeding Fake Data

You can use Additional tools to capture screenshot of the app, which is important for submitting your app to the marketplace. You can feed fake map as well as accelerometer data.



Calling Delegates without Blocking UI Thread

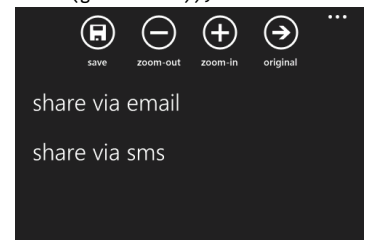
Every *UIElement* (basically all elements are derived from this type eventually) has a *Dispatcher* property, which lets you invoke a delegate without blocking the UI thread from which it is being invoked:

```
void cell_Click(object sender, RoutedEventArgs e)
{
    Dispatcher.BeginInvoke(() => MessageBox.Show(gameResult));
}
```

AppBar

A default project template will give you Application Bar commented out in *MainPage.xaml*:

```
<phone:PhoneApplicationPage.ApplicationBar>
  <shell:AppBar IsVisible="True" IsMenuEnabled="True">
    <shell:AppBarIconButton IconUri="/Images/appbar_button1.png" Text="Button 1"/>
    <shell:AppBarIconButton IconUri="/Images/appbar_button2.png" Text="Button 2"/>
    <shell:AppBar.MenuItems>
      <shell:AppBarMenuItem Text="Item 1"/>
      <shell:AppBarMenuItem Text="Item 2"/>
    </shell:AppBar.MenuItems>
  </shell:AppBar>
</phone:PhoneApplicationPage.ApplicationBar>
```



Launchers and Choosers

There are classes that allow to prompt user for range of input types and use it in your app. Many of them also allow to use default phone tasks, such as SMS compose Photo chooser. The following example allows user to take photo which your app can directly use. In this case, we have rendered it in an Image element:

```
private readonly CameraCaptureTask _CameraCaptureTask = new
CameraCaptureTask();

public MainPage()
{
    InitializeComponent();
    _CameraCaptureTask.Completed
        += _CameraCaptureTask_Completed;
}

private void ApplicationBarMenuItem_Click(object sender,
EventArgs e)
{
    _CameraCaptureTask.Show();
}

void _CameraCaptureTask_Completed(object sender, PhotoResult
e)
{
    if(e.TaskResult == TaskResult.OK)
    {
        var bmp = new BitmapImage();
        bmp.SetSource(e.ChosenPhoto);
        image1.Source = bmp;
    }
}
```

Another example, which opens phone's default SMS compose screen with pre-filled values:

```
new SmsComposeTask {Body = "Hello, SMS!", To =
"+880101010101"}.Show();
```

NavigationService

Your app will not live in a single page. It will obviously need multiple pages, so navigating around them is a very necessary aspect to address. Pages on Windows Phone work much like browser. When you click on Back after you have visited a link in a browser, it will take you to the originating page. Both browser and Windows Phone's navigation system keep track of the visited links in a stack, and in both cases Back and Forward work the same way. Let us take a look how you can navigate from one page to another along with parameters:

```
NavigationService.Navigate(new Uri("/Forecast.xaml?City=" +
city.Text, UriKind.Relative));
```

Now in the Forecast.xaml, see how it is being picked up:

```
protected override void OnNavigatedTo(NavigationEventArgs
e)
{
    var city = this.NavigationContext.QueryString["City"];
}
```

Making a Web Request

You can use *WebClient* as well as *WebRequest* for this, but let us look at the latter, the more difficult one here:

```
// initialize a new WebRequest
var woeidRequest = (HttpRequest) WebRequest.Create(new
Uri(url));

// set up the state object for the async request
var woeidState = new State {AsyncRequest = woeidRequest};

// start the asynchronous request
woeidRequest.BeginGetResponse(HandleWoeidResponse, woeid
State);
.....
private void HandleWoeidResponse(IAsyncResult asyncResult)
{
    // get the state information
    var woeidState = (State) asyncResult.AsyncState;
```

```
var woeidRequest = woeidState.AsyncRequest;
// end the async request
woeidState.AsyncResponse = (HttpWebResponse) woeidRe
quest.EndGetResponse(asyncResult);

try
{
    // get the stream containing the response from the
    // async call
    var state = woeid
        State.AsyncResponse.GetResponseStream();
    var xmlState = XElement.Load(state);
    .....
}

public class State
{
    public HttpRequest AsyncRequest { get; set; }
    public HttpWebResponse AsyncResponse { get; set; }
}
```

IsolatedStorage

You can save files for your app. You can even serialize your objects to files and retrieve and deserialize when necessary. Let us see this very useful helper class for such operations:

```
internal sealed class StorageHelper
{
    public static void DeleteFile(string fileName)
    {
        using (var appStorage = IsolatedStor
ageFile.GetUserStoreForApplication())
        {
            appStorage.DeleteFile(fileName);
        }
    }

    // Example: StorageHelper.Save("student.dat", student);
    public static void Save<T>(string fileName, T thing)
    {
        using (var appStorage = IsolatedStor
ageFile.GetUserStoreForApplication())
        {
            using (var file = appStorage.OpenFile(fileName,
                FileMode.Create))
            {
                using (var writer = new StreamWriter(file))
                {
                    var serializer = new XmlSerializer
                        (typeof(T));
                    serializer.Serialize(writer, thing);
                    writer.Close();
                }
                file.Close();
            }
        }
    }

    // Example: var student = StorageHelper.Read<Student>
    ("student.dat");
    public static T Read<T>(string fileName) where T : new
        ()
    {
        T thing;

        using (var appStorage = IsolatedStor
ageFile.GetUserStoreForApplication())
        {
            using (var stream = new IsolatedStor
ageFileStream(fileName, FileMode.OpenOrCreate,
                FileAccess.Read, appStorage))
            {
                using (var reader = new StreamReader
                    (stream))
                {
                    var serializer = new XmlSerializer
                        (typeof(T));
                    thing = reader.EndOfStream ? new T() :
                        (T)serializer.Deserialize(reader);

                    reader.Close();
                    stream.Close();
                }
            }
        }

        return thing;
    }
}
```

Submitting to Marketplace

Complete guideline: <http://bit.ly/9mWNla>