Get Phone Location with Windows Phone 8

This article explains how to get Phone Location in Windows Phone 8. This is a new feature introducted with the last version of Windows Phone OS

Introduction

Windows Phone 8.0 offers 2 ways for tracking the position of the phone:

- Continuously
- Single-Shot

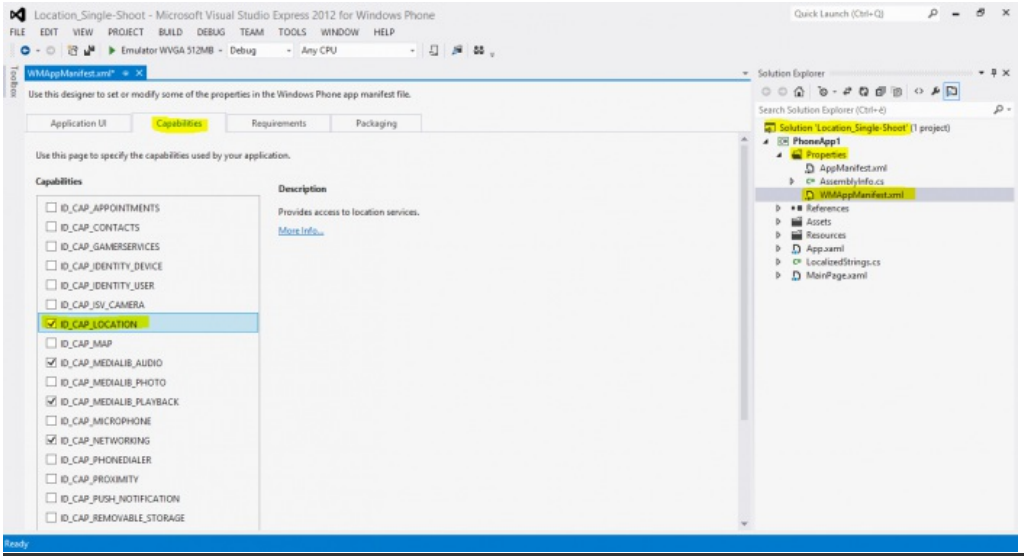
While the first one allows you to continuously tracking and so get a more complete information, it also drains the user's battery more than the second choice, i.e. single-shot location. Because of this, continuous tracking should only be used for apps that require this kind of function. If you want to use an app which only needs a user's location at the current time, the Single-Shot option is the best choice.

In this topic both of the methods are explained and their source code is also shared.

Warning: This functionality requires to work the Location API, and so your app must include in the ID_CAP_LOCATION capability in your app manifest file. Omit this operation will make your app throw an exception when you deploy it during development and will cause your app to fail ingestion when you submit it to the Windows Phone Store.

How to include the ID_CAP_LOCATION capability in app manifest file

1. Create a new **Windows Phone App** and then select **Windows Phone 8.0** as platform you want to target
2. Go to **Solution Explorer**, expand the **Properties** folder, and then double-click **WMAppManifest.xml**
3. Click on the **Capabilities** tab of the manifest designer, select the check box next to `ID_CAP_LOCATION`



Ask for user consent prior to using the location APIs

Another important thing that must be done both in Continuously location, and in Single-Shot, is to ask to the user if your app can use his position.

All apps should obtain user consent prior to using the location APIs.

A way to do it is by creating a consent prompt to allow the user to opt out of allowing your app to access their location, and to save his choice. It is suggested to make this appear when the user launch the app. So we will add it in the `OnNavigatedTo(NavigationEventArgs)` method of the `MainPage` class. First of all, you have to check if the user is allowing your app to use his location. If yes, the method returns immediately, otherwise a popup will be shown to the user, and the result saved to a key in `ApplicationSettings`.

Let's write and analyze the code step by step

1. First of all override the original `OnNavigatedTo(NavigationEventArgs)` method, by writing this code

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
//Do Your Work here
}
```

2. Next we have to check if the user has given his permission to share his location with our app. If yes, the apps return immediately.
3. All the code you will write from now must be put inside the `OnNavigatedTo(NavigationEventArgs)` method we wrote before. Let's assume that the key you have choose for saving the user choice is called `LocationConsent`

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

```
if (IsolatedStorageSettings.ApplicationSettings.Contains("LocationConsent"))
    {
    //User already gave us his agreement for using his position
    if ((bool)IsolatedStorageSettings.ApplicationSettings["LocationConsent"] == true)

        return;
else
{
//Ask for the user consent
}
    }
else
{
//Ask for the user consent
}

}
```

4. Last thing you have to do is to create a consent prompt for ask user allowing to use his Location. If he will allowing we will save his choice, otherwise no, and the next time he will open the app, he will see consent prompt again

```
 protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
if (IsolatedStorageSettings.ApplicationSettings.Contains("LocationConsent"))
    {
    //User already gave us his agreement for using his position
    if ((bool)IsolatedStorageSettings.ApplicationSettings["LocationConsent"] == true)

        return;
        //If he didn't we ask for it
    else
    {
        MessageBoxResult result =
                    MessageBox.Show("Can I use your position?",
                    "Location",
                    MessageBoxButton.OKCancel);

        if (result == MessageBoxResult.OK)
        {
            IsolatedStorageSettings.ApplicationSettings["LocationConsent"] = true;
        }
        else
        {
            IsolatedStorageSettings.ApplicationSettings["LocationConsent"] = false;
        }

        IsolatedStorageSettings.ApplicationSettings.Save();
    }
    }

    //Ask for user agreement in using his position
else
{
MessageBoxResult result =
            MessageBox.Show("Can I use your position?",
            "Location",
            MessageBoxButton.OKCancel);

        if (result == MessageBoxResult.OK)
        {
            IsolatedStorageSettings.ApplicationSettings["LocationConsent"] = true;
        }else
        {
            IsolatedStorageSettings.ApplicationSettings["LocationConsent"] = false;
        }

        IsolatedStorageSettings.ApplicationSettings.Save();
    }
}
```

How to continuously track the phone's location for Windows Phone 8

Warning: This way of tracking position consumes more power and will drain the battery faster.

1. The first operation you have to do is to add the right using statements to your **MainPage.xaml.cs**.
   - For doing this open the **MainPage.xaml.cs**, by clicking on it in the **Solution Explorer**
   - Add the following code to the top of the file

```
using System.Threading.Tasks;
using Windows.Devices.Geolocation;
using System.IO.IsolatedStorage;
```

2. Then create a variable for the Geolocator object. It is also suggested to use a boolean variable for checking the tracking status of the app (you will see it better in the example). Both the variables must be added inside the MainPage class definition. Here is the code:

```
Geolocator geolocator = null;
bool trackingStatus = false;
```

3. If you want to enable the tracking you have to:
   - Inizialize the **Geolocator** object

```
geolocator=new Geolocator();
```

- Set **Accuracy** and **Movement Throdeshold**
  Accuracy could be set to *High* or *Default*. Second one is better for power consumption.
  Movement Threshold is the distance of required movement, in meters, for the location provider to raise a `PositionChanged` event.
  Let's see the code:

```
geolocator.DesiredAccuracy = PositionAccuracy.High;
geolocator.MovementThreshold = 100;
```

4. Last, but not least, event handlers for the `StatusChanged` and `PositionChanged` events are registered.

```
geolocator.StatusChanged += geolocator_StatusChanged;
geolocator.PositionChanged += geolocator_PositionChanged;
```

5. Now, it has to be write the 2 event handlers that we mentioned before
   - For the `StatusChanged` you can copy the code below. With this handler your app will be able to communicate to the user the status of the tracking operation. But since this event handler is not called on the UI thread, you must use `BeginInvoke(Action)` to run any code that modifies the UI. Take a look to the code

```
void geolocator_StatusChanged(Geolocator sender, StatusChangedEventArgs args)
{
    string status = "";

    switch (args.Status)
    {
        case PositionStatus.Disabled:
            // the application does not have the right capability or the location master switch is off
            status = "location is disabled in phone settings";
            break;
        case PositionStatus.Initializing:
            // the geolocator started the tracking operation
            status = "initializing";
            break;
        case PositionStatus.NoData:
            // the location service was not able to acquire the location
            status = "no data";
            break;
        case PositionStatus.Ready:
            // the location service is generating geopositions as specified by the tracking parameters
            status = "ready";
            break;
        case PositionStatus.NotAvailable:
            status = "not available";
            // not used in WindowsPhone, Windows desktop uses this value to signal that there is no hardware capable to acq
            break;
        case PositionStatus.NotInitialized:
            // the initial state of the geolocator, once the tracking operation is stopped by the user the geolocator moves

            break;
    }
 /*This part write the status to the UI element StatusTextBlock*/
    Dispatcher.BeginInvoke(() =>
    {
        StatusTextBlock.Text = status;
    });
}
```

   - The handler for the `PositionChanged` event is raised whenever the phone has moved a distance greater that the movement threshold from the previous location. Again, this event handler is not called on the UI thread, so you must use `BeginInvoke(Action)` to run any code that modifies the UI. It is very simply

```
void geolocator_PositionChanged(Geolocator sender, PositionChangedEventArgs args)
{
    Dispatcher.BeginInvoke(() =>
    {
        LatitudeTextBlock.Text = args.Position.Coordinate.Latitude.ToString("0.00");
        LongitudeTextBlock.Text = args.Position.Coordinate.Longitude.ToString("0.00");
    });
}
```

6. When you stop tracking position you have to remove the 2 event handlers and to set the object `Geolocator` to `null`

```
geolocator.PositionChanged -= geolocator_PositionChanged;
       geolocator.StatusChanged -= geolocator_StatusChanged;
       geolocator = null;
```

How to get the phone's current location for Windows Phone 8 (Single Shot)

1. The first operation you have to do is to add the right `using` statements to your **MainPage.xaml.cs**.
   - For doing this open the **MainPage.xaml.cs**, by clicking on it in the **Solution Explorer**
   - Add the following code to the top of the file

```
using System.Threading.Tasks;
using Windows.Devices.Geolocation;
using System.IO.IsolatedStorage;
```

2. After you have to create a variable for a variable for the Geolocator object.
   The variable must be added inside the MainPage class definition. Here is the code:

```
Geolocator geolocator = null;
```

3. If you want to enable the tracking you have to:
   - Inizialize the `Geolocator` object

     ```
     geolocator=new Geolocator();
     ```

   - Set `DesiredAccuracyInMeters` property - it indicates the desired accuracy in meters for data returned from the location service. Let's see the code:

     ```
     geolocator.DesiredAccuracyInMeters = 50;
     ```

4. Then we can call the `GetGeopositionAsync()` method. This method attempts to obtain the phone's current location. It does this asynchronously so that the UI thread is not blocked while the location is obtained. This is called inside a `try` block in case any exceptions are thrown.

This method, as said before, works asynchronously so that the UI thread is not blocked while the location is obtained. You can use the `await` operator to place code after the asynchronous call that will be executed after the call finishes. This requires this handler method to be declared `async`. Because calls made using await are guaranteed to return on the thread from which they were called, and this await call was made from the UI thread, the code is able to access and modify the UI directly when the call returns.

```
private async void OneShotLocation_Click(object sender, RoutedEventArgs e)
{
//Check for the user agreement in use his position. If not, method returns.
    if ((bool)IsolatedStorageSettings.ApplicationSettings["LocationConsent"] != true)
    {
        // The user has opted out of Location.
        return;
    }

    Geolocator geolocator = new Geolocator();
    geolocator.DesiredAccuracyInMeters = 50;

    try
    {
        Geoposition geoposition = await geolocator.GetGeopositionAsync(
             maximumAge: TimeSpan.FromMinutes(5),
             timeout: TimeSpan.FromSeconds(10)
            );

//With this 2 lines of code, the app is able to write on a Text Label the Latitude and the Longitude, given by {{Icode|geopos
        LatitudeTextBlock.Text = geoposition.Coordinate.Latitude.ToString("0.00");
        LongitudeTextBlock.Text = geoposition.Coordinate.Longitude.ToString("0.00");
    }
//If an error is catch 2 are the main causes: the first is that you forgot to include ID_CAP_LOCATION in your app manifest.
//The second is that the user doesn't turned on the Location Services
    catch (Exception ex)
    {
        if ((uint)ex.HResult == 0x80004004)
        {
            StatusTextBlock.Text = "location is disabled in phone settings.";
        }
        //else
        {
            // something else happened during the acquisition of the location
        }
    }
}
```

All the pieces are complete now.

Helpful Resources

If you want an example on how to use it, my advice is to give a look to the downloadable example available in this topic

- Single-Shot Location Media:SingleShotLocExample.zip
- Continuous Location Media:Continuous Location.zip

If you want more info about resorces used in this wiki article i remand you to this links:

- http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.geolocation.geolocator
- http://msdn.microsoft.com/en-us/library/System.IO.IsolatedStorage.IsolatedStorage.aspx