Location based operations using Bing Maps

This article describes the basic tasks needed for applications using location based services on Windows Phone.

Introduction

Some basic functions needed for location based services are described in this article:

- Fetching the current location
- Reverse-geocoding this location to get an address (that can be displayed to the user)
- Geocoding addresses to get map locations
- Calculating a route between locations
- Displaying all these on a map

This article is complemented by a sample Media:BingMapsSample.zip that shows how to fulfill these tasks in a WP app.

Note: This article covers Windows Phone 7.5, but the same code should work with version 8.0. Windows Phone SDK Version 8 however contains contains new APIs in the Windows.Devices.Geolocation namespace that extend the capaibilities and are compatible with WinRT apps. The article will be revised soon to cover the new features. An overview of the new APIs is given in the Windows Phone Developer Blog articles Updating your Windows Phone Location code to use WinRT, Acquiring a single Geoposition in Windows Phone 8 and Geoposition advanced tracking scenarios for Windows Phone 8.

Preliminaries

Please add references to

- Microsoft.Phone.Controls.Maps
- Microsoft.Phone.Controls.Toolkit
- System.Device
- System.Device.Location

and then add service references to Bing Maps's SOAP geocoding service and routing service:

- http://dev.virtualearth.net/webservices/v1/geocodeservice/geocodeservice.svc?wsdl
- http://dev.virtualearth.net/webservices/v1/routeservice/routeservice.svc/mex

Fetching the current location

Everything you need to determinate the current location is available in every Windows Phone without the need to connect online. You have to create a GeoCoordinateWatcher instance and attach to the `StatusChanged` and `PositionChanged` events. It's important to set a movement threshold that defines how often a new notification is given. Too many notifications reduce the performance/battery charge level. Then you call `Start()` which is an asynchronous call that returns immediately, but the results are returned in the above mentioned events.

See also this Location Overview for Windows Phone and How to: Get Data from the Location Service for Windows Phone.

```
private void GetCurrentLocation()
   { // get current location by asking WP location service
   if (watcher == null)
      {
      watcher.StatusChanged += new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
      watcher.PositionChanged += new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
      watcher.Start();
      }
   }

void watcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
   { // status of the location service has changed - stop if disabled
   if (e.Status == GeoPositionStatus.Disabled)
      {
      watcher.Stop();
      watcher.StatusChanged -= new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
      watcher.PositionChanged -= new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
      watcher = null;
      }
   }

void watcher_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
   { // the current position has changed - we need only one so we stop the service now
   watcher.Stop();
   watcher.StatusChanged -= new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
   watcher.PositionChanged -= new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
   watcher = null;
    //...//
   }
```

Reverse-geocoding the current location

Since we have referenced the Bing Maps SOAP geocode service, you get access to geocoding and reverse-geocoding functions. You can use geocoding to get street addresses from location coordinates and reverse-geocoding to get an address from a coordinate. An online connection is needed to call these

(C) Copyright Nokia 2013. All rights reserved.

functions.

The code illustrated below initializes the service and then sets up a call to reverse geocoding with the current location to get the current address. To create the request, you need to give your Bing Maps credentials. You can get credentials for non-commercial Windows Phone apps free of cost from the Bing Maps Portal.

See also Getting a Bing Maps Key and GeocodeServiceClient.ReverseGeocode Method.

Please don't forget to enter your Bing Maps Key into the `BingMapsKey` variable in MainPage.xaml.cs.

```
// initialize geocode service
GeocodeServiceClient gc = new GeocodeServiceClient("BasicHttpBinding_IGeocodeService");

// reverse geocode the location to get a displayable address
gc.ReverseGeocodeCompleted += new EventHandler<ReverseGeocodeCompletedEventArgs>(gc_ReverseGeocodeCompleted);
ReverseGeocodeRequest rreq = new ReverseGeocodeRequest();
rreq.Credentials = new Credentials();
rreq.Credentials.ApplicationId = BingMapsKey;
rreq.Location = e.Position.Location;
gc.ReverseGeocodeAsync(rreq);
```

Geocoding an address

Now a call to the service is used for geocoding to get a coordinate for the address of the german Nokia branch office. Again, the credentials are needed to initialize.

Both geocoding calls are issued concurrently to the SOAP Geocode Service. It is not determined which one of these calls completes first.

See also GeocodeServiceClient.Geocode Method.

```
// concurrently geocode an address in Germany
destination = e.Position.Location;
gc.GeocodeCompleted += new EventHandler<GeocodeCompletedEventArgs>(gc_GeocodeCompleted);
GeocodeRequest req = new GeocodeService.GeocodeRequest();
req.Credentials = new Credentials();
req.Credentials.ApplicationId = BingMapsKey;
req.Query = "Balcke-Dürr-Allee 2, 40882 Ratingen, Germany";
gc.GeocodeAsync(req);
```

Calculating a route

In order to calculate a route you again need to initialize the Route Service and then set up the request. The request parameters this time consist of the Bing Maps key, options and a list of waypoints. The waypoints in this case are the geo-locations of the Nokia branch your current location. In the options the `RoutePathType` is important, because it tells the service to return all intermediate route points so that a street route can be displayed.

After the route is completed, the `CalculateRouteCompleted` event is called. There a `RouteResult` is returned. The `RouteResult` contains a summary with travel distance and duration and a detailed description of the route.

```
// initialize call
RouteServiceClient rc = new RouteServiceClient("BasicHttpBinding_IRouteService");
rc.CalculateRouteCompleted += new EventHandler<CalculateRouteCompletedEventArgs>(rc_CalculateRouteCompleted);
RouteRequest req = new RouteRequest();
req.Credentials = new Credentials();
req.Credentials.ApplicationId = BingMapsKey;
req.Options = new RouteOptions();
req.Options.Mode = TravelMode.Driving;
req.Options.RoutePathType = RoutePathType.Points;
req.UserProfile = new BingMapsSample.RouteService.UserProfile() { DistanceUnit = BingMapsSample.RouteService.DistanceUnit.Kilomet

// setup the waypoints
Waypoint wpStart = new Waypoint();
wpStart.Location = start;
wpStart.Description = "Nokia DE";
req.Waypoints = new System.Collections.ObjectModel.ObservableCollection<Waypoint>();
req.Waypoints.Add(wpStart);
Waypoint wpEnd = new Waypoint();
wpEnd.Location = destination;
wpEnd.Description = "Me";
req.Waypoints.Add(wpEnd);

// display start and destination with pushpins
map1.Children.Clear();
Pushpin p = new Pushpin();
p.Location = wpStart.Location;
p.Content = wpStart.Description;
p.Background = new SolidColorBrush(Colors.Red);
map1.Children.Add(p);
p = new Pushpin();
p.Location = wpEnd.Location;
p.Content = wpEnd.Description;
p.Background = new SolidColorBrush(Colors.Blue);
map1.Children.Add(p);

// start the route calculation
rc.CalculateRouteAsync(req);
```

Displaying in a map

To display the route on a map,

(C) Copyright Nokia 2013. All rights reserved.

- A map element is added to the MainPage xaml code.
- Before calling the route service, two pushpins are added to the map for the start and destination locations.
- When the route calculation is complete, the result is analyzed and a `MapPolyline` object is constructed that contains the route points.
- The last operation is to zoom the map view to the bounding rectangle of the route.

```
void rc_CalculateRouteCompleted(object sender, CalculateRouteCompletedEventArgs e)
   {
   // get summary of the route
   RouteResult rr = e.Result.Result;
   tbDist.Text = rr.Summary.Distance.ToString() + " km";
   tbTime.Text = (rr.Summary.TimeInSeconds / 60).ToString() + " min.";

   SolidColorBrush brItinerary = new SolidColorBrush(Colors.Green);
   SolidColorBrush brLeg = new SolidColorBrush(Colors.Red);
   List<GeoCoordinate> pts = new List<GeoCoordinate>();

   // create a line object from the route points
   MapPolyline line = new MapPolyline();
   line.Stroke = brItinerary;
   line.StrokeThickness = 5;
   line.Locations = new LocationCollection();
   foreach (var pt in rr.RoutePath.Points)
      line.Locations.Add(pt);

   // show the line in the map and zoom to its bounds
   map1.Children.Add(line);
   map1.SetView(rr.Summary.BoundingRectangle);
   }
```

Download code example

- The example illustrated in this article is available for download here.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.