

John C. Zablocki
Development Lead, MagazineRadar
Consulting Developer, I'mOK
Hartford Code Camp
2011-06-18

building location based apps with windows phone 7

agenda

- Why Location?
- WP7 Location Services
- GeoCoordinateWatcher
- GPS Emulator
- Bing Maps Silverlight Control
- SimpleGeo
- Questions?

why care about location?

- VC investment in the mobile space was over \$6 billion in 2010 - <http://yhoo.it/hxKW7l>
- Smartphones are now outselling PCs - <http://huff.to/efHzZZ>
- Social is growing and going mobile (i.e., Facebook Places, Google Places)
- Location is fueling mobile's growth. Foursquare grew 3400% last year - <http://4sq.com/fWeNaO>

windows phone 7

Location Services



windows phone 7 location servi

- All Windows Phone 7 devices provide access to location services in one of three ways
 - GPS
 - Cell Towers
 - WiFi
- Native code layer sits on top of the hardware and determines device location through available resources.
- Managed interface is layered over the native code layer to provide location information to managed apps.

ces wp7 location best practic

- Apps must weigh the need for accuracy of location against power consumption
 - The GPS is generally the most accurate source of location data on a Win7 device
 - WiFi and cell towers are less accurate than GPS, but consume less power
- Regardless of source, location services tax device resources
 - Turn on location services only as needed

es geocoordinatewatcher and i

- IGeoPositionWatcher interface defines events for position and status changes updates from location hardware
- GeoCoordinateWatcher class implements IGeoPositionWatcher and adds properties for setting desired level of accuracy and minimum threshold before location change notifications are fired

geopositionwatcher

- GeoCoordinateWatcher defines:
 - Constructor for setting the GeoPositionAccuracy argument (enum of High or Default)
 - PositionChanged event
 - MovementThreshold to set the number of meters before PositionChanged event is raised
 - StatusChanged event to handle when location information becomes available
- Need to reference System.Device assembly to get access to the managed location API

geocoordinatewatcher example

```
private IGeoPositionWatcher<GeoCoordinate>
_watcher;

_watcher = new GeoCoordinateWatcher();
            (_watcher as GeoCoordinateWatcher)
            .MovementThreshold = 20;

_watcher.PositionChanged += new
            EventHandler<GeoPositionChangedEventArgs
            <GeoCoordinate>>(watcher_PositionChanged);
_watcher.StatusChanged += new
            EventHandler<GeoPositionStatusChangedEventArgs
> (_watcher_StatusChanged);
```

geocoordinatewatcher example

```
void watcher_PositionChanged(  
    object sender,  
    GeoPositionChangedEventArgs  
        <GeoCoordinate> e) {  
  
    //do something  
}
```

```
void _watcher_StatusChanged(  
    object sender,  
    GeoPositionStatusChangedEventArgs e) {  
  
    //do stuff  
}
```

gpsemulatorclient

- Location services are not available in the WP7 emulator that ships with WP7 developer tools
- To address this limitation, MS developers have released a GPS emulator - <http://bit.ly/eoHSRb>
 - Includes a small WPF application that communicates with the WP7 emulator to provide location data
 - Includes a WP7 dll for use during dev

gpsemulatorclient example

```
#define GPS_EMULATOR
#if GPS_EMULATOR
    _watcher = new
    GpsEmulatorClient.GeoCoordinateWatcher();
#else
    _watcher = new
    System.Device.Location.GeoCoordinateWatcher();
    (_watcher as GeoCoordinaateWatcher).MovementThreshold =
    20;
#endif
_watcher.PositionChanged += new
    EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(
    watcher_PositionChanged);
_watcher.StatusChanged +=
    new EventHandler<GeoPositionStatusChangedEventArgs>
    (_watcher_StatusChanged);
```

bing maps

Silverlight Control



windows phone 7 bing map cont

- Silverlight Bing Maps control exists in the Microsoft.Phone.Controls.Maps assembly
- Initialize map by setting Center property to GeoCoordinate in XAML or code
- ZoomLevel (1-23) and Mode (Aerial, Road) properties allow for changing map appearance
- ZoomBarVisibility property toggles visibility of the zoom (+ or -) buttons on the map
- MapItemsControl allows for data (template) driven child controls (e.g., Pushpins)

simplegeo

API and Object Model



rols simplegeo service overvie

- SimpleGeo provides services for applications that wish to include location services.
 - SimpleGeo Context for information, such as population density, weather, etc.
 - SimpleGeo Storage for building scalable applications with location cloud stored location data
 - Places API for point of interest data
- SimpleGeo Places and SimpleGeo Context are free, with premium support available
- SimpleGeo Storage is in private beta and will be a paid solution

w simplegeo restful api

- API is available through a REST interface
- Authentication is performed via special case of OAuth – Two-Legged
 - OAuth is simply used for authenticating the request, not allowing access on behalf of a third party (since there is none)
- All responses conform to valid HTTP status codes (i.e., querying a place that does not exist will yield an HTTP status code 404 and creating a record will result in a 201 status).

simplegeo places api

- Primary use case for Places API is to retrieve a list of nearby places
- Each call to the Places API requires using 1 of 4 ways to pass the starting point:
 - Lat/Long pair
 - A human-readable, US address
 - An IP address
 - None of the above (user IP is used)
- Only GET requests are supported
- Optional parameters (q, category, radius, and num) allow for filtered results.
- Result is a GeoJson document

simplegeo sample api calls

<http://api.simplegeo.com/1.0/places/37.7645,-122.4294.json?q=Starbucks>

<http://api.simplegeo.com/1.0/places/37.7645,-122.4294.json?category=Restaurant>

<http://api.simplegeo.com/1.0/places/37.7645,-122.4294.json?category=Restaurant&q=diner>

simplegeo object model

- **Features** represent real-world places, such as businesses, regions or states
- **Handles** uniquely identify a feature
 - With a known handle, SimpleGeo's Feature API may be called without an account
- **Point** refers to a coordinate on a map
- **Polygon** is an area bounded by a closed path of points

simplegeo features

- Places API returns a collection of GeoJSON encoded features.
- Each feature has information about:
 - Long/Lat Coordinates
 - SimpleGeo Handle
 - Properties such as address and phone and feature taxonomy (type and category)

simplegeo geojson result

```
{
  "geometry":{
    "type":"Point",
    "coordinates":[
      -122.421583,
      37.795027
    ]
  },
  "type":"Feature",
  "id":"SG_5JkVsYK82eLj26eomFrl7S_37.795027_-122.421583@1291796505",
  "properties":{
    "province":"CA",
    "city":"San Francisco",
    "name":"Bell Tower",
    "tags":[],
    "country":"US",
    "phone":"+1 415 567 9596",
    "href": "http://api.simplegeo.com/1.0/features/SG_5JkVsYK82eLj26eomFrl7S_37.795027_-122.421583@1291796505.json",
    "address":"1900 Polk St",
    "owner":"simplegeo",
    "postcode":"94109",
    "classifiers":[
      {
        "type":"Food & Drink",
        "category":"Restaurant",
        "subcategory":""
      }
    ]
  }
}
```

simplegeo and .net

Calling SimpleGeo from WP7 Apps

simplegeo .net client

- SimpleGeo provides official clients and SDKs for Objective-C, Android, Java and Python
- Ruby, PHP and .NET all rely on community build efforts
- Original .NET client SimpleGeo.NET
 - Abandoned in April 2010
 - No longer compatible with current API
 - Was written in VB.NET!
- Recent effort by Jörg Battermann looks promising, though still early in its development
- At this time, no current client project exists for .NET that is in working order!

simplegeo .net client hack

- A quick and dirty solution is available at <http://bit.ly/g6Qdjr>
- Basic idea, create a POCO object graph to match the GeoJSON document
 - Feature, FeatureCollection
 - Classifier, Geometry, Properties
- Use [Hammock for REST](#) for OAuth and service consumption
- Use [JJson.NET](#) to deserialize GeoJson to FeatureCollection graph

simplegeo .net client code

```
//TODO: allow multiple category and query search
public FeatureCollection GetNearbyPlaces(double latitude, double longitude,
    string query = null) {

    var path = string.Format("places/{0},{1}.json", latitude, longitude);

    path += !string.IsNullOrEmpty(query) ? "?q=" + query : "";

    var request = new RestRequest() { Path = path };
    var response = this.Request(request);

    if (response.StatusCode == HttpStatusCode.OK) {
        var jsonObj = JObject.Parse(response.Content);
        return JsonConvert
            .DeserializeObject<FeatureCollection>(response.Content);
    } else {
        //TODO: raise intelligent exceptions
        return new FeatureCollection() { Features = new List<Feature>() };
    }
}
```

simplegeo .net wp7 client code

```
public void GetNearbyPlaces(double latitude,
    double longitude, string query = null) {

    var path = string.Format("places/{0},{1}.json", latitude,
        longitude);

    path += !string.IsNullOrEmpty(query) ? "?q=" + query : "";

    var request = new RestRequest() { Path = path };
    BeginRequest(request, RequestComplete);
}
```

simplegeo .net wp7 client code

```
public void RequestComplete(RestRequest request,           RestResponse
response, object userState) {

    FeatureCollection fc = null;

    if (response.StatusCode == HttpStatusCode.OK) {
        var jsonObj = JObject.Parse(response.Content);
        fc = JsonConvert
            .DeserializeObject<FeatureCollection>(response.Content);
    } else {
        fc = new FeatureCollection() {
            Features = new List<Feature>()
        };
    }
    if (null != RequestCompleteEventHandler) {
        RequestCompleteEventHandler(fc);
    }
}
```

simplegeo .net wp7 mvvm code

```
public void LoadData(string location, string taxonomy) {
    Client client = new Client("KEY", "SECRET");
    client.RequestCompleteEventHandler += new
Action<FeatureCollection>(client_RequestCompleteEventHandler);

    var locationSplit = location.Split(',');
    double latitude = double.Parse(locationSplit[0]);
    double longitude = double.Parse(locationSplit[1]);

    if (null != BeginDownload) {
        BeginDownload();
    }
    client.GetNearbyPlaces(latitude, longitude, taxonomy,
"restaurant", 300);
}
```

simplegeo .net wp7 mvvm code

```
public void client_RequestCompleteEventHandler
    (FeatureCollection obj) {

    LocalStateContainer.Dispatcher.BeginInvoke(() => {
        foreach (var feature in obj.Features) {
            if (feature.Properties.Phone.Trim() ==
MISSING_PHONE_NUMBER) {
                feature.Properties.Phone = null;
            }
            Features.Add(feature);
        }
        if (null != EndDownload) {
            EndDownload();
        }
    });
}
```

Links

- <http://dllHell.net> - my blog
- <http://www.CodeVoyeur.com> - my code
- <http://www.linkedin.com/in/johnzablocki>
- <http://twitter.com/codevoyeur>
- <http://www.simplegeo.com>
- <http://bitbucket.org/johnzablocki/codevoyeur-samples>
- <http://about.me/johnzablocki>

Questions?
