Pogo Rent 3000 Part 1

# Background

Pogo Rent 3000 is a fictional company that hires pogo sticks round the world. Through a serious of articles will show how to add GIS functions to its website. For start the company wants to show a basic map with great Pogo parks in northern Sweden.

The aim of the articles is to give and idea how to build robust GIS applications build on .Net and Bing Maps.

The aim of this first article is to describe the implementation of Pogo Rent 3000. The idea of this project is to provide a solid implementation of a GIS project based on Bing Maps. The techniques and framework used are well known and provide a robust foundation for GIS applications.

We strongly believe that using standard formats for communication with GIS backend is something that is of importance and cannot be stated enough. Showing an example an endpoint that simulates a Web Feature Service (WFS from OGC, see <http://www.opengeospatial.org/standards/wfs>) or included in the code as a GeoServiceController.

# Application structure

The application is structured the simplest way possible and providing a robust platform to extend the system. We build a classic 3-tier application.



## UI Tier

On top we have Bing Maps V8, it is one of the best map widget out there as today. It provides a straight forward, documented API. More over a set of spatial function like measuring and a set of modules. A module is functionality you can add on to the web client. For example, GeoJSON support. You can also write your own modules.

## Service Tier

As service tier we have a Asp.NET Web API. This tier is boosted with Net Topology Suit (NTS, <https://github.com/NetTopologySuite/NetTopologySuite>) which provide a rich set of GIS operations. NTS is great for adding spatial functions to the application. It also provides GeoJSON support. With this in place we have the possibility to serve OGC formats.

## Data Tier

In this example we use a single text file as data source. The source is a GeoJSON file. We could of course easily just ask for the GeoJSON file directly through a GET request from the client but for demo purpose we actually read and parse the file with NTS. The data tier also has an interface,

Reading GeoJSON with NTS is really simple, here is a snippet from the application:

var reader = new NetTopologySuite.IO.GeoJsonReader();

var features = reader.Read<FeatureCollection>(data);

*Data* is the GeoJSON. It will parse the json into an object which we can spatial filter, manipulate compare or do other operations.

# Wrap up

With the techniques and framework in place, it is pretty straight forward to implement. The implementation is as simple as possible and it provides possibility to extend the PogoRent3000 system. To summarize, building a GIS application has the same structure as any 3-tier application.

POGO RENT 3000 PART 2

In the first part we described how to build for a robust GIS system. In this part we’ll move on to static maps and modules

# Static Maps

Static maps are great. In many occasions you just need to provide an image. You really don’t need an interactive map. Sometimes you just want to show a location, or a route between locations. For this purpose, static maps are great.

So where does static maps fit in our solution. On our web site! We want to show spots where it is great to pogo. It is really to embed a static map. Just treat it as any image! For example

<img src="http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/-28.014407569005286,153.42029571533203/12?mapSize=200,200&key={BING KEY} />

The above will include an image on a web page that is 200x200px in size. Since it is an image it can be treated as an image. For example, put them in a slider. It is a bit tricky to get the URL correct, here is the documentation on MSDN <https://msdn.microsoft.com/en-us/library/ff701724.aspx>. And a 3rd party configuration tool is available online here <http://staticmapmaker.com/bing/>.

In the example site, we have just included static maps as images since it fits our purpose of showing where to pogo.

# Modules

Modules is a pluggable technique to add functions to Bing Maps. In the demo site the GeoJSON module being used. There are a numerous modules, for a complete list see here, <https://msdn.microsoft.com/en-us/library/dd877180.aspx>. There is also an open source project for modules on Codeplex, the project is originally for Bing Maps v7 control, the good news is most of them works with v8 control. There is a compability list here <https://bingmapsv7modules.codeplex.com/wikipage?title=Module%20Compatibility%20with%20Bing%20Maps%20V8>.

In the public web site for this article the Custom Inbox Control are used. It means you are able to click on parks and the information available.