Pogo Rent 3000 Part 1

# Background

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

Pogo Rent 3000 is a fictional company that hires pogo sticks round the world. As a consultant, I am asked to help implement GIS functionality to their web site and to where ever it gives value. Through a couple of posts I will show how to address this in a pragmatic way. For start the company wants to show a basic map with great Pogo parks in northern Sweden.

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.

I strongly recommend using standard formats for communication with GIS backend is really importance and cannot be stated enough. The application also showing an example of 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. OGC formats and protocol gives flexibility and maintainability.

# Application structure

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



## UI Tier

On top there is Bing Maps V8, it is one of the best map client 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 extend the web client with. For example, with GeoJSON, (<http://geojson.org>) support. The small script that is needed is, for convenience, written in TypeScript. GeoJSON is an OGC standard since 2016. This means we can add in other datasets from other sources without worrying about the format as long as it is GeoJSON.

With the GeoJSON module it is super easy to parse the JSON from the server.

let primitives = Microsoft.Maps.GeoJson.read(geoJSONtext) as Array<Microsoft.Maps.IPrimitive>;

## Service Tier

As service tier I 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. I could of course easily just accessed GeoJSON file directly through a GET request from the client but for demo purpose I actually read and parse the file with NTS. The data tier also have an interface. The idea with the interface is that it should be easy to add other data sources, for example SQL Server.

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 is not different than building any 3-tier application.