

Internet GIS: Google Maps and Mashups

introduction [both lecture and lab + public workshop]

As we proceed, you'll probably notice that my PowerPoint is a little thin. As data practitioners, we should all be familiar with Edward Tufte [TUFF-tee]. Not only do I agree with his criticisms of PowerPoint, but I'm also personally tired of attending lectures that are little more *than* PowerPoint presentations. I'm tired of the fast slide flipping and the stupid animations. So, hopefully you won't mind too much that I don't have a bullet list for everything.

Having said that, I've set up a website to support my GIS Day activities in Milwaukee. All of the workshops/tutorials are available on the site, as are handouts, which can be downloaded and printed from the site [including this talk]. I also created a forum to facilitate mashup development at neogeography.net.

So, here's the plan. We'll do a bit of theory [i.e. you'll have to listen to me for a while] and then we'll get to work building our own Google mashup. An outline of what I'll be talking about is on the wall now.

Here's what I assume and how it'll go—for the most part, I assume you're familiar with web technologies. This means a conceptual understanding of protocols such as <http://>, <ftp://>, client- vs. server-side, etc. I assume you have some coding knowledge of HTML, XHTML, XML, CSS, and JavaScript. I'm going to be throwing out a lot of jargon and acronyms during the lecture and workshop. Some I might explain, others I might not. So, if you have any questions, feel free to stop me. If I think your question is very basic [note this is not the same as stupid, because we all know there's no such thing as], I might poll the group to see if it's worth going into detail. I'll try to answer everything, though. If not during the talk, then afterwards or on the forum.

Oh yeah, I have a tendency to talk fast, so let me know if I need to slow down or repeat something or if I need to go into more detail.

So, what is a mashup? What are people hoping to learn? What's everyone's background?

definitions [both lecture and lab + public workshop]

A term borrowed from pop music [the combination (usually by digital means) of the music from one song with the a cappella from another], a mashup is nothing more than a website or Web 2.0 application that uses content from more than one source to create a completely new service. The functions that power and the content used in mashups are typically sourced from a third party via an application programming interface [API]. Content for mashups can also be retrieved using feeds [RSS, Atom], JavaScript [AJAX], and XML [Wikipedia].

Given the current availability of simple and lightweight APIs, mashups are relatively easy to implement, requiring minimal technical knowledge and little overhead. Basically, they make application development fast, cheap, and out of control.

Since we'll actually be developing in this class, instead of just pasting in some produced code [the one-minute mashup], it might help to go a little more in depth on APIs. An API is the interface that a computer system or application provides in order to allow requests for services to be made of it by other computer programs and/or to allow data to be exchanged between them. We'll be communicating with this interface, working with the objects, methods, and properties that

it has defined. So, even if you're familiar with the language an API is written in, you really need to get familiar with the API calls. Google provides online documentation of their API.

There are a number of different ways to talk to APIs, including REST, SOAP, AJAX, etc. Google Maps uses AJAX. AJAX provides a slick interface, because it's client side. Pages can be smoothly updated without the standard call-and-response flash as the server reloads the page. Unfortunately, client-side code can also be very slow. Normally you shouldn't notice it, but if you have a lot of applications open, your Google Maps app is going to drag. Also, if you load a lot of overlays [markers and polylines] at once, the app will drag.

So, that's the brief technical bit. Any questions?

some options :: mapping apis

While there are too many APIs to cover and we're obviously here to talk about Google Maps, I think it's good to remember that Google is not our only choice. Aside from Google, all of the major content providers offer a free mapping API, including Yahoo, MSN, and MapQuest. [Note that we have only worked with Google and MapQuest.]

According to Schuyler Erle of Mapping Hacks, "early on ... there was a huge gap between Google Maps and the rest. ... The big three ... have basically converged, and their map display APIs look more or less alike, implementation details aside." Supposedly, Yahoo has tried to offer web services for things Google has been unwilling or unable to do, including Flash map APIs in JavaScript and ActionScript and REST services for geocoding, map image delivery, traffic reporting, and so on. APIs less elegant, but offer a lot more functionality. [Maybe next year we'll be doing a workshop on Yahoo maps?]

His conclusion: "Hackers still love Google, as they always have, and Google Maps continues to demonstrate a sensitivity to their sensibilities. With the addition of their JavaScript and particularly their REST APIs, Yahoo offers more in terms of hackability, but we still haven't seen much in the way of uptake—one presumes this is because of the popular conception of Yahoo. Similarly, MSN offers a map API that is solid but not uniquely compelling, all the less so because of the non-commercial use limitation required to keep it competing from MapPoint."

So, which API to choose? Depending on the project, it might be better to go with a service like Yahoo or MapQuest. In general, I prefer Google because I believe the hype about their corporate culture. Personally, I avoid Yahoo due to the China scandal.

Another option to consider is ArcWeb Services. ArcWeb Services expose true GIS functionality, but they are not free. ArcWeb Public Services are free for a year, but they only offer a subset of the full functionality. I've worked briefly with ArcWeb Public Services. Their geocoding service is more accurate than geocoder.us, but I could never get a map to display using PHP and SOAP.

some options :: other

Besides the mapping APIs, though, there is a ton of other APIs out there worth remembering. Want images or video on your map? Use Flickr or YouTube. Want to display the weather based upon the map coordinates? Try weather.com.

The limit to a mashup is your imagination and ability to program.

approaches

There are two primary ways to get a mashup on your web page: using an existing service or library to generate the code for you or code your own. Depending on your needs, the former

might be faster. Bloggers and web site designers might be interested in Platial's MapKit. Using MapKit is fast and simple, offers some customizability, and does have some functionality, but I don't think any GIS person will be satisfied with it.

The other approach [coding our own] takes more work and requires a bit of programming experience, but it's the *only* solution for custom application development.

limitations + legalities

Aside from the obvious limitations of generated code, there are limitations [sometimes and sort of] even when we code our own. Beside general browser limitations and security policies, generally, we're limited to what the API makes possible [though there generally are workarounds]. For instance, Google didn't originally offer geocoding. To get around this, I built a mashup that would geocode using the ArcWeb public services and then display using Google Maps [these are mashups after all].

Google can:

- geocode [now]
- do polygons
- do map tiles

Google cannot:

- do routing
- do any real analysis

There are also legal restrictions. Most, if not all, mashups require "signing" a service agreement. We'll discuss Google's in a minute, when I talk about getting started with Google Maps.

Finally, there are some inherent limitations to mashups in general. The same things that make APIs great [you didn't have to build all that functionality] are also their weaknesses. If the host of the API ever goes away, your application doesn't work. You have to constantly be ready for upgrades to the API that may not be backwards compatible.

mashups, gis, + ppgis

Okay, we're almost out of the theory and ready to start coding, but a few thoughts. While the mapping APIs are cool, easy to use [for the most part], and free, they are not GIS. Most are not structured to handle anything but displaying point data well, and even then they don't provide many analysis tools. Most also do not support routing or any type of vector or polygon analysis.

However, what they do supply is coordinate-based mapping, geocoding, and base maps. So while the mapping APIs aren't a GIS out of the box, they can be built as one. Most of my tutorials are an attempt to design and implement a GIS via Google Maps, where a GIS is defined as capable of displaying, analyzing, managing spatial data and associated attributes.

I also feel that working with Google Maps [actually all APIs] is truly PPGIS. We're no longer reliant on expensive proprietary software. APIs encourage modular design to overcome their inherent limitations, which means our code should be more flexible and forward compatible. We're able to give our nonprofit and community-based groups the ability to share and control their data.

advanced topics :: google maps and flash, postgresql

This class/workshop is really focused on implementing Google Maps solely via JavaScript and XML data stores. While we're not going to work with Flash during this workshop

and I don't want to overwhelm anyone, there's the possibility of some really slick development with Flash. In fact, I think the future of open-source will be a data-driven backend with a Flash front-end [embedded Google Maps of course].

There's a Google Maps component for Flash 9 out there somewhere, but I can't find it.

VGMap is a library developed by Eyebeam R&D that adds vector drawing capability to the Google Maps API. Essentially, VGMap ties the Google Maps API to Flash, while also providing a few ActionScript libraries and samples to simplify the process of drawing over the map correctly. While VGMap is supposedly fairly easy to use, building a Flash application that takes data and plots it on a map requires some amount of understanding of georeferencing/geocoding, cartographic projections, geographic data file types, and/or all the other nonsense us mapping/GIS geeks obsess about.

More information about VGMap can be found at <http://vgmap.eyebeamresearch.org/>.

While MySql has been the workhorse for web-development, I believe the future of Google Maps and open-source GIS development will be led by PostgreSql and PostGIS: "PostGIS is a project which adds support for geographic objects in PostgreSQL, allowing it to be used as a spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension." [postgreSQL]

Ok, on to the tutorials.

getting started with google maps [in general]

First, an overview of what we need. Access to a web server is primary. If we're going to do any real development with Google Maps, our server should be able to serve active content [PHP, ASP, ColdFusion] and we'll also need access to a database server. Worst-case scenario, an Access database on the web server is possible, but not a production solution. In essence, a standard web setup [preferably LAMP].

We'll also need some data to do anything interesting. These data can exist in other applications, be accessed via API, in a text document or database, or in an XML document. For our tutorials, we'll be using static XML documents.

Finally, we need an API key. For the purpose of this workshop, we'll use mine, but you can sign up for your own at <http://www.google.com/apis/maps/signup.html>. API keys are free, but are only good for a single directory on the webserver. You also have to agree with the terms and conditions [legal restrictions I mentioned above].

getting started with google maps [this workshop]

[browser, notepad, ftp client slide]

Sources

Erle, Schuyler. Web Map API Roundup. <http://mappinghacks.com/2006/04/07/web-map-api-roundup/#more-120>. Retrieved 5 November 2006.

PostgreSQL. <http://www.postgresql.org/about/>. Retrieved 14 November 2006.

Wikipedia. Mashup. http://en.wikipedia.org/wiki/Mashup_%28web_application_hybrid%29. Retrieved 5 November 2006.