GEOG863::MASHUPS FOR GIS PROFESSIONALS

(FALL 2011)

DEANNE LUNDIN

FINAL PROJECT SUMMARY REPORT

Overview: Create a mashup that allows a user to select an event category from a dropdown menu which is dynamically updated to contain only those categories for which there are currently events. The selected category will populate a sidebar and display markers for the event locations on the map. The data source is The Village Voice Events Calender.

Breaking the task into its separate functions and sequencing these gave me the final initial list:

- **Scrape data from the original web page (Village Voice) using PHP**
- **Format the retrieved data into well-formed XML**
- **Geocode the locations for these events**
- **Write the geocoded venues into a file that can be used to check whether the venue has been geocoded already before geocoding**
- **Create an HTML document that holds the map, selection box, and sidebar**
- **Call the PHP script to populate the dropdown menu**
- **On selection, populate the sidebar with the selected array of values**
- **Display markers on the map for the selection made**
- **Get everything to work as separate functionality before combining**

LINK to the final php script as a text file here:
(http://personal.psu.edu/dxl295/GEOG863/VillageVoiceToXMLGeo.txt)

Future development:

- Worked primarily in Firefox. Among other things Firefox does not currently implement getElementById on XML documents which cost quite a lot of time trying to fix since my attempt to use it kept returning null.
- Different color markers for each category (and a concise legend) so that it's easy to tell what kind of event you're seeing on the map when "all" is loaded.
- Add a thumbnail of the event's promo image to the infoWindow.
- Use external CSS. It's a messy script in the div section.
- I don't expect this app to get used by anyone except a few friends who live in Brooklyn. The Village Voice does provide links to the venue's street address and a map inset. What it doesn't provide is what this app aimed to do: nothing but the info and a context for all related events. If you're out to dinner and the show is cancelled, what else is nearby? For that kind of functionality, if I was serious, I'd want to develop it for smart phones.

Problems

- Extracting data from the VV source.
- PHP testing. MySQL installation and decision not to create another database project but work with dynamic data stream.
- Debugger(s) and learning to use them.
- Creating well-formed XML from uncertain input.
- Creating and managing multiple arrays
- Managing time issues
- Disappearance of the Google API V2 function for clearing overlays.

Making a Dynamic Mashup

> Sometimes I feel like the maid. I just cleaned up this mess!
>
> (Mr. Incredible)

Mashups are viewed by many serious programmers as the shallow end of the pool where we've all got floaties. It's true that anyone can jump in. (Here I am.) Making original structures of functional beauty and style takes rather more and there are some truly inspired applications out there.

Creating a working mashup that does something more than aggregate new feeds is fraught with unexpected chasms and rockslides. An even better metaphor would be caving. There you are down in a deep rift and a flash flood roars in. If you're creating an enterprise web application that will persist longer than this sentence because it's functionally gorgeous and everyone wants to use it, you won't be just "spend some time coding then sit back and watch money roll in" as one bitter commenter posted ("Mashups: who's really in control?" http://www.zdnet.com/blog/web2explorer/mashups-whos-really-in-control/128).

Because they're intuitive—and fun—ways to use geographic information, mashups are popular enough to have spawned yet another product line for developers who like to make tools. You can jump in with little or no coding experience and use templates and wizards to make a "mashup." But of course it will suffer from everything a non-custom product usually does: it will exhibit behaviors you don't expect; it will probably have just enough functionality to get you into trouble. And when something does go wrong you won't know why and certainly won't be able to fix it. You'll be waiting for the developers who do know how to code to get back to you.

I'm never happy waiting for tech support to get back to me. That never goes away, but I want to be able to formulate better questions, do better research, and feel more capable of experimenting because I have some fluency with basic programming concepts and problem-solving. I'd like to be able to wear the geek shirt ("My Apps Are Well Developed," maybe). Right now mine still reads "Sounds Like Geek To Me."

So it looked like time to work more intensively with javascript, making functions and figuring out how things actually worked—particularly *when* things actually worked, which turned out to be a major conceptual hurdle for me. Having expert tutorial help meant I eventually could work through the various problems in the time I had to do it. Without that, this project would have failed. I could have done it by myself in 3 weeks or a month, but not in less than 2, which is what I ended up with once I'd worked through application difficulties, like getting MySQL and the correct connector installed and configured, which then turned out to be irrelevant for this project once I defined what it was I wanted to do.

Project Highlights

In past projects I've spent more time on the front end working my strategy out in clear English. In this case, once I realized I didn't want to do another database-driven project but try working with bringing in a live, changeable data stream, the basic outline was simple: scrape the page using php, reformat and output as xml to the main html document, and use html and JavaScript to create the display and its functionality. Well, okay. And how do you do that? I needed to review everything I already "knew" to establish explicit understanding of how each scripting language worked, how and why you'd use each, and which ones play well together (or really, really don't).

EasyPHP was far easier to use than the access provided by the university. Having a local server for testing and experimenting was crucial—so much of what I had to do was either fussy (using the strpos and substr technique to work through a string input to get only the substring I needed). Without good clear understanding of what data I was actually getting, this project would have been impossible.

Timing is everything

Even when it looked like things were in order, debugging revealed that I wasn't getting data back when and how expected. And one solution to a timing issue remains something I still don't understand. Here's the note from my script:

```
//This is an unexpected hack job.  The DIV element I get back is null.
// I'm guessing the HTML isn't completely built yet, despite
// that I do nothing before initialize(), which is called from
// the <BODY> onLoad event.  Why wouldn't the <DIV> element be
// retrievable??? Don't know...
// Wait until the <DIV> element that encloses the <SELECT> dropdown
// is available.
var DivElem = document.getElementById("mydiv");
while(DivElem == null) {  // repeat until the div is available.
   // wait 1/5 second, then call the simplest anonymous function ever written.
   var t=setTimeout(function(){}, 200);
   // these never appear in the console log.
   console.log('Waiting for the div to be ready');
   DivElem = document.getElementById("mydiv");
}
```

Fortunately for me my tutor is not only a highly experienced programmer, he's so good at debugging he's pretty much the firm's "fixer" (Autodesk). I really had no idea how to solve this one.

The other issues mainly were a matter of the complexity of what had looked like a relatively feasible task. As one unexpectedly tricky matter, getting the xml well-formed kept me busy trying to account for all the special formatting in the names of events. The script would work nicely for a few days while I went on to other tasks and then it would fail, thanks to another Village Voice update. I was actually grateful to them for thinking of more ways to plague the parser (my favorite so far is the title "<3 attack" followed by three different groups concatenated by + signs). But parse failures were plentiful in the beginning and took a lot of time to work through carefully. In the process I was introduced to regular expressions in PHP (which I now think of as PCP). And munching my way along a sizeable string retrieving substrings, while it's a process I understand conceptually, was surprisingly challenging in keeping track of where you were and what you'd just done.

More complicated was getting the geocoding process working. There was a simple question of whether the geocoding service would return anything for a "premise" query. It would have saved me a lot of coding because the venues which I was mapping didn't include addresses on the main page I was scraping. I did however already have the links Village Voice provides to take the user to a page with further information for each venue. Testing the "premise" convinced me that while yes, it would work, results were uncertain at best. Here's what I posted about this:

```
Example:

I have a place for which I know the exact street address (though not
immediately available programmatically without scraping yet another
page).  Entering the place name in Google Maps will return the named
venue along with its listed exact address ("The Glasslands Gallery, 289
Kent Street, Brooklyn, NY").

Sending that same query to geocoding svc (URL method) returned either
nothing (with its locality of Brooklyn, NY specified) or uncertain
results (when using only region NY).  And the first return was
somewhere in Manhattan, which knocks out the hope of just taking the
zero-th item. In Deannese, the new scripting language so many are
using, expressed as

     ?!FOO!-->[an array of confused and ambiguous results]

EXACT STREET ADDRESS sent to geocoding svc returned the appropriate
latlon for my place (tested in reverse via Google maps by inputting the
latlon).
```

But at least my question was answered. Yes, I did have to scrape yet another page, associate all the data, stuff it back into the working code and hope nothing breaks.

I also didn't want to be geocoding every venue every time the webpage was loaded. I set up a small secondary .csv file to cache the data and then had to write functions to test whether the venue name being inspected was a)in that little database yet; and if so b)did I already have the latlon values. And every new piece of functionality was another opportunity to load the webpage and see white space instead of data.

The HTML I'd quite looked forward to. While I don't think I'm terribly fluent in anything, at least I'd used JavaScript/HTML/CSS before. I had code snippets. I knew where to go for information and I (mostly) understood what I read. But what looked to be a very easy matter of HTML <form> tags turned

out to be more problematic: I wasn't using a static xml file as a data source. And applying the options only to the select element's innerHTML didn't work on Internet Explorer. I couldn't fill in the <option value>tags as the function stood. I was going to have to build the html myself, somewhat in the way the sidebar's html is made.

Keeping track of how arrays needed to be formed and when was probably the greatest challenge for me. Data had to be brought in and associated with variables, parsed into xml (thanks to Mike for his library here), a select dropdown created, and even when the dropdown populated properly, I'd failed to associate the selection event with the sidebar array for that category.

There were a number of tools to help me do this (see table below).

Particularly useful:	**Problem**
• EasyPHP	None (but see below)
• Xdebug w/Notepad ++	Learning to use the debugger wasn't difficult but did take some practice. More time.
• Mozilla's Firebug (esp. console.log)	Don't always get expected behavior. Ended up doing a lot of restarting of my server, the html document, and Firebug itself.
• Other methods of getting data where I could see it (fwrite and fflush in PHP code)	

About the course

Like the other GIS courses in the PSU Program, this felt like an excellent introduction. For experienced programmers, it adds function to an already well-developed set of skills. For someone who is essentially novice, each project presented increasingly difficult problems to solve, many of which were solvable in a clumsy cut-and-paste-code-snippets sort of way. While that's always a good place to start (and I would have been grateful for a wider array of examples) my need to form an intellectual matrix on which I could build made me put in some long hours reading, experimenting, trying to understand, and not least, getting help from an experienced programmer. I wanted to be able to solve things like a programmer, not like someone who was going to be buying the Template Tool. In the end I was introduced to many basic programming concepts (the difference between procedural and object-oriented programming or how to use regular expressions, for instance) that left me wishing I had the kind of programming background you only get from an intensive undergraduate degree program at a competitive university. I agree completely with the well-trained programmers who survey the burgeoning field of newbies thinking, Whatever happened to Real Programming? And a glance through the most popular developers' forums reveals a huge number of rookies who don't even know how to post a well-made question much less code—but then asking better questions does tend to be the result of better education. Trying to solve problems at all

is sometimes what I've had to settle for. What I want is being able to solve problems in more than one way.

I've considered my PSU education as proof of concept—do I really want to work in GIS? Do I want to develop enterprise applications? Is the MGIS really for me? With the CGIS I'm taking away a set of barely-fledged skills that are probably just enough to get me into serious jeopardy with a firm conviction that you never get past occasional all-nighters and it still looks like fun to me.

That makes it a lot like writing. Research. Collaborative projects. Teaching. Learning how to use new technologies.

So that would be a yes.