Project 3: Google Newsworthy

due by 5:30 P.M. ET on Monday, 21 April 2008

Goals.

- Implement a mashup!
- Leverage Ajax and RSS.
- Read the news.

Recommended Reading.

- Google Map API Concepts
  http://code.google.com/apis/maps/documentation/

- Google Map API Reference
  http://code.google.com/apis/maps/documentation/reference.html

- RSS 2.0 Specification
  http://cyber.law.harvard.edu/rss/rss.html

- Procedural Programming in MySQL
  http://www.quest-pipelines.com/newsletter-v6/0105_D.htm
  http://www.quest-pipelines.com/newsletter-v6/0205_D.htm

- Using PHP/MySQL with Google Maps
  http://code.google.com/support/bin/answer.py?answer=65622&topic=11364

- Using Debugging Tools with the Google Maps API
  http://code.google.com/support/bin/answer.py?answer=87133&topic=11364

Academic Honesty.

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (as by the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's classroom) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this class that you have submitted or will submit to another. Moreover, submission of any work that you intend to use outside of the course (*e.g.*, for a job) must be approved by the staff.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss projects with classmates, but you may not share non-trivial amounts of code (*e.g.*, entire files). If in doubt as to the appropriateness of some discussion, contact the staff.

You may turn to the Web for instruction beyond the course's lectures and sections and for solutions to technical problems, so long as you cite the origin of any code that you encounter and incorporate into your own work (as with CSS, PHP, or XHTML comments). Failure to cite the origin of any such code may be considered academic dishonesty.

All forms of academic dishonesty will be dealt with harshly.

Grades.

Your code (CSS, JavaScript, PHP, SQL, XHTML, *etc*.) will be evaluated along the following axes.

- **Correctness.** To what extent does your code meet our specifications?
- **Design.** To what extent is your code written clearly, efficiently, elegantly, and/or logically?
- **Style.** To what extent is your code commented and indented, your variables aptly named, *etc*.?

cs75.net.

❑ Surf on over to:

http://cs75.net/surveys/3/

Please take a few minutes to provide candid answers to the survey's questions. Although you may be prompted to authenticate, your answers will be anonymized upon submission. We will know that you took the survey, but we will not know which answers are yours.

panel.cs75.net.

❑ Surf on over to http://panel.cs75.net/, log in, and follow the link to **Subdomain Management** under **Your Account**. On the page that appears, add a subdomain called **project3** to your domain. Not only will the panel automatically create A records for project3.domain.tld and www.project3.domain.tld for you, where domain.tld is your domain, it will also create a subdirectory called project3 in your public_html directory. All of your work for this project must ultimately reside within that subdirectory.

❑ Return to your panel's home page and select **Password Protected Directories**, also under **Your Account**. On the page that appears, click **Find a Directory to Password Protect**. You should then see the contents of your public_html directory. In the row containing your project3 subdirectory, click **Protect** in the column labeled **Action**. On the page that appears, check the box next to **Protection Enabled**, and then fill out the rest of the form. Any username and password are fine, as is any value for **Protected Directory Prompt**, which is just an aesthetic detail. When done, click **Save**. The panel will proceed to create (or edit) at least two files for you: ~/public_html/project3/.htaccess, which tells Apache to password-protect the directory, and ~/domains/domain.tld/.htpasswd/public_html/project3/.htpasswd, which contains your choice of username and password, separated by a colon, with the latter encrypted. Had you not a convenient panel, you could also have just created files like those by hand.

❑ Return to your panel's home page and select **MySQL Management**, also under **Your Account**. On the page that appears, click **Create a new Database**. Provide a name, username, and password for your database, then click **Create**. Make note of all details on the page that appears! You must use that database for this project.

project3.domain.tld.

❑ Your mission for this project is to implement a mashup that integrates Google Maps with Google News with a MySQL database containing 42,492 zip codes. But first, some inspiration!

If you've a friend who owns a Nintendo Wii (that's connected to the Internet), see if you can invite yourself over there for "homework purposes." Ask your friend to start up the **News Channel**, then click **National News** with the Wiimote, then click any of the articles, then click **Globe** in the

screen's bottom-right corner. Notice how you can spin the Earth by clicking **A** and dragging, thereby revealing stacks of articles from different cities and geographic areas. And by zooming in and out with **+** and **–**, you can reveal more or fewer stacks. By clicking a stack's icon, you can then read local news.

If you haven't said friend, watch this instead, paying close attention between 01:04 and 01:50:

http://www.youtube.com/watch?v=uO6J8ryTKYk

The challenge ahead isn't to implement precisely that interface but the sprit thereof in the (largely) two-dimensional world of Google Maps! Specifically, your mashup should present users with a Google Map, next to which should be a form. Upon submitting a zip code via that form, users should whisked away on the map to that zip code's location, which should be overlaid with markers representing news articles related to that area. Clicking a marker should trigger a GInfoWindow with the titles of and links to one or more articles about the area. However, much like the Wii's News Channel, your mashup should include markers for articles from the surrounding area (*i.e.*, other zip codes) as well!

Alright, so where to find all this data?

❑ First surf on over to Google Maps at http://maps.google.com/. Input a zip code like **02138** into the page's form and click **Search Maps**. You should find yourself whisked away to a familiar location. Well that was easy. Notice, though, that the page's URL did not change. I wonder how they did that!

Now input **28.42, -81.58** instead. Perhaps you'd rather be there?

It seems Google Maps understands zip codes as well as latitude and longitude. Interesting...

❑ Now surf on over to Google News at http://news.google.com/. Click the link to **Advanced news search** at top right, and notice how you can "Return only articles about a local area." Input a zip code like **02138**, and you should see news local to Cambridge. Notice now the link to **RSS** on the page's left-hand side. Click it, and you should find yourself at a URL like the below.

http://news.google.com/news?svnum=10&as_scoring=r&ned=us&as_drrb=q&as_qdr=m&as_mind=28&as_minm=2&as_maxd=30&as_maxm=3&geo=02138&aq=f&output=rss

Within that otherwise cryptic string should be a familiar value. Hm, that could be useful...

❑ Alright, clearly there's a way to whisk a user away to a particular point on a map, using zip codes or latitude and longitude. But Google News expects only zip codes, so how to find zip codes that are proximal to another? It'd be nice if we had a big list...

Surf on over to http://www.teamredline.com/zc/.

Not bad for $5.00! We'll pick up the tab, though. Go ahead and download the zip file we bought:

http://cs75.net/projects/zipcodes.zip

Inside that archive you'll find a number of files, among them ZipCodes.txt and ZipCodes.xls. Open up either to get a sense of the data. Note that the files pair most zip codes with geographic coordinates (presumably the zip codes' "centers") as well as with cities, states, and abbreviations. We've taken the liberty of converting that data into a MySQL table, which you should download as well:

http://cs75.net/projects/zipcodes.sql

You should import that table into your own database (as via phpMyAdmin's **Import** tab).

❑ Okay, you now know the geographic locations of zip codes. Given one zip code, though, how do you find others near it?

Thanks to GoonDocks.com,[1] here's a MySQL function that'll compute the distance in miles between two latitudes and longitudes:

```
DELIMITER $$

DROP FUNCTION IF EXISTS `GetDistance`$$

CREATE FUNCTION `GetDistance`(
 lat1 numeric(9,6),
 lon1 numeric(9,6),
 lat2 numeric(9,6),
 lon2 numeric(9,6)
) RETURNS decimal(10,5)
BEGIN
 DECLARE x decimal(20,10);
 DECLARE pi decimal(21,20);
 SET pi = 3.14159265358979323846;
 SET x = sin( lat1 * pi/180 ) * sin( lat2 * pi/180  ) + cos(
 lat1 *pi/180 ) * cos( lat2 * pi/180 ) * cos( abs( (lon2 * pi/180) -
 (lon1 *pi/180) ) );
 SET x = atan( ( sqrt( 1- power( x, 2 ) ) ) / x );
 RETURN ( 1.852 * 60.0 * ((x/pi)*180) ) / 1.609344;
END$$

DELIMITER ;
```

---

[1] Our thanks to Gabe Summer for his January 22 post!
 http://www.goondocks.com/blog/08-01-22/zip_code_radius_search_using_mysql.aspx

And here's a procedure that'll return a nearby zip codes, given one zip code and radius:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `GetNearbyZipCodes`$$

CREATE PROCEDURE `GetNearbyZipCodes`(
    zipbase varchar(6),
    range numeric(15)
)
BEGIN
DECLARE lat1 decimal(5,2);
DECLARE long1 decimal(5,2);
DECLARE rangeFactor decimal(7,6);
SET rangeFactor = 0.014457;
SELECT latitude,longitude into lat1,long1 FROM tbl_zipcodes WHERE zipcode =
zipbase;
SELECT B.zipcode
FROM tbl_zipcodes AS B
WHERE
 B.latitude BETWEEN lat1-(range*rangeFactor) AND lat1+(range*rangeFactor)
 AND B.longitude BETWEEN long1-(range*rangeFactor) AND long1+(range*rangeFactor)
 AND GetDistance(lat1,long1,B.latitude,B.longitude)  <= range;
END$$

DELIMITER ;
```

Go ahead and execute both blocks of code via phpMyAdmin's **SQL** tab. You can then invoke the procure in PHP with code like:[2,3]

```
$result = mysql_query("CALL GetNearbyZipCodes('02138', 100)");
```

The above should return a result set of zip codes within 100 miles of Cambridge. That's a lot of zip codes (and soon news)! Know, incidentally, that Google's API models distance in meters.

❑   Now sign yourself up for a Google Maps API key:

http://code.google.com/apis/maps/signup.html

Be sure to specify http://project3.domain.tld/ as your URL.[4]

---

[2] Note that you might not be able to call the stored procedure within phpMyAdmin.
[3] In order to call stored procedures, you'll need to connect to your MySQL database with code like:

  mysql_connect(HOST, USERNAME, PASSWORD, 0, 65536);

  Alternatively, you can use PHP's mysqli extension.

[4] For local development, you might also need to sign up for a key for http://127.0.0.1/, but be sure to configure your site with your key for http://project3.domain.tld/ for submission.

❑ Okay! It's time to start mashing all of these things together. The overall design and aesthetics of this mashup are ultimately up to you, but we require that it meet some requirements. All other details are left to your own creativity and interpretation.

Feature Requirements.

❑ Your mashup should present users with a Google Map, next to which should be a form. Upon submitting a zip code via that form, users should whisked away on the map (without the page itself reloading) to that zip code's location, which should be overlaid with markers representing news articles related to that area. Clicking a marker should trigger a GInfoWindow with the titles of and links to one or more articles about the area. However, much like the Wii's News Channel, your mashup should include markers for articles from the surrounding area (*i.e.*, other zip codes) as well!

❑ Your map should grow to fill the user's window, while still leaving room for that form, much like Google Maps itself.

❑ For memory's sake, you should only display news (and thus markers) for zip codes that are currently within view, given the map's zoom level. You should not place markers outside the confines of the map's GLatLngBounds. And you should clear any markers that suddenly become (thanks to dragging, zooming, or provision of a new zip code) outside of those GLatLngBounds.

❑ For performance's sake, you should not display news for every zip code within the map's GLatLngBounds, particularly if the map is zoomed out quite far. (For instance, if the entirety of the United States fits within the map's current GLatLngBounds, you should not query Google for news on 42,492 zip codes.) We leave it to you to decide which zip codes to show; we expect you to show at least "a few." Pseudorandomness or other heuristics are probably your friend.

❑ To find articles for some zip code, you must query via Ajax a PHP file within your domain that itself queries Google News for an RSS (or Atom) feed and returns the results (in some form) to your mashup via XML or JSON. To find zip codes (and thus more articles) proximal to the user's input, you must query that $5.00 table.

❑ For Ajax calls, you may use any off-the-shelf library (*e.g.*, YUI's Connection Manager or Google's GXmlHttp namespace) or roll your own.

❑ We leave it to you to decide how many articles to associate with a given zip code. For small numbers of articles, it's fine to display one marker per story. Given that a zip code maps to just one latitude and longitude, though, be sure not to place multiple markers completely on top of each other. (Perhaps pseudorandomness or some other heuristics can help you lay things out.) Alternatively, particularly for large numbers of articles, it's fine to display one marker per zip code.

❑ For performance's sake, you should not necessarily display links to all available articles for some zip code. (For instance, as of 30 March 2008, 02138 boasted 51,608 recent articles!) Filter as you see fit.

❑ Links to articles should open in new windows so that the user's original window remains at your mashup.

❑ Your mashup must somehow handle all possible errors (*e.g.*, invalid zip codes, zero articles, *etc.*) gracefully. Under no circumstances should users encounter JavaScript runtime errors.

Technical Requirements.

❑ Your site must live at http://project3.domain.tld/, where domain.tld is your own domain.

❑ All PHP files must be chmod'd 600.

❑ Your XHTML must be valid (or "tentatively" valid), unless some feature of your site requires otherwise (for the sake of some browser); explain in XHTML comments any intentional invalidities. Your XHTML should also be as pretty-printed as possible. Your CSS need not be valid.

❑ Your JavaScript and PHP must be extensively commented and be as pretty-printed as possible.

❑ You may use a WYSIWYG editor to generate XHTML and/or CSS that you would like to use in your site.

❑ If you incorporate or adapt snippets of code from the Web into your project, cite the code's origins with comments.

❑ If you incorporate images from the Web into your project, cite the images' origins with XHTML comments.

❑ Your website must appear and behave the same on at least two major browsers, namely:
    ❑ Firefox 2.x
    ❑ Internet Explorer 7.x
    ❑ Opera 9.x
    ❑ Safari 3.x

Submission.

❑ Once done with your site, put together a readme at http://project3.domain.tld/readme/. Treat this readme as your opportunity not only to explain but to justify your design decisions. Tell us with which two (or more) browsers we should evaluate your site. And give us an overall sense of how your site works (*e.g.*, tell us which files do what). But still be succinct; keep this readme to just a few paragraphs in length.

❑ By this project's deadline, submit your website by SSHing to your domain and executing the command below in order to submit the contents of your ~/public_html/project3/ directory.

submit project 3

Thereafter, follow any on-screen instructions until you receive visual confirmation of your work's successful submission. You may re-submit as many times as you'd like, but take care not to re-submit after the project's deadline, lest your work be deemed late.