

Paper 052-2010

Get to Your Points: Using SAS® to Build Google Maps

Ted Conway, Chicago, IL

ABSTRACT

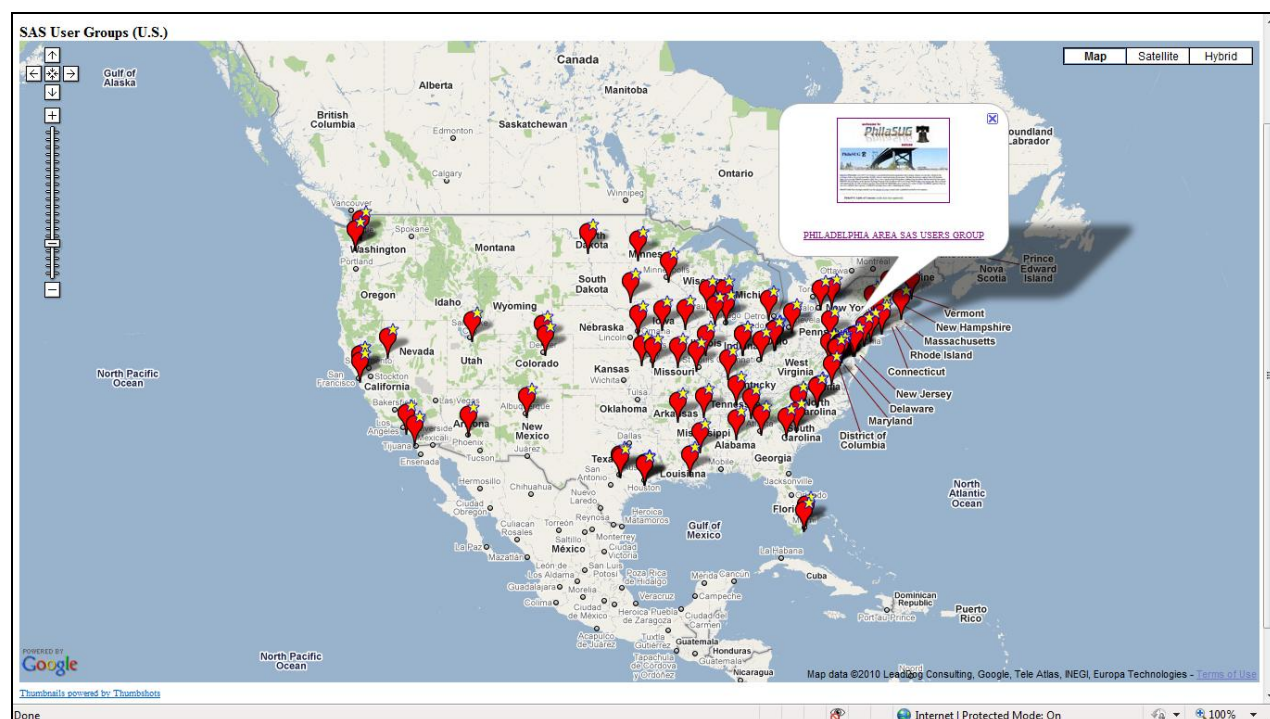
Ever want to present your data on an interactive Google map? If so, SAS® can make your life easier! This paper explains how to use the Google Maps API to overlay SAS User Group information--converted to a suitable XML format using SAS, of course--on a Google map of the U.S.

INTRODUCTION

If you've used SAS/GRAPH, you know that certain data lends itself very well to presentation on a map. And now, thanks to APIs from companies like Google and Microsoft, it's relatively easy to overlay all kinds of data and information on interactive online maps. But don't throw away your SAS yet – prepping the data is 90% of the battle, and you'll find SAS invaluable even when it's not being used to present the final results!

HOW ABOUT AN EXAMPLE?

In our example, we're going to use SAS and the Google Maps API to present SAS User Group information on a Google map of the United States:



GOOGLE MAPS API – CUSTOM MAP

In the Google API-based map above:

- ✓ Markers are displayed to show the location of Local, Regional, and Special Interest SAS User Groups
- ✓ Infowindows with SAS User Group website previews pop-up when map markers are moused-over
- ✓ SAS User Group websites are launched in a new window when map markers or website previews are clicked

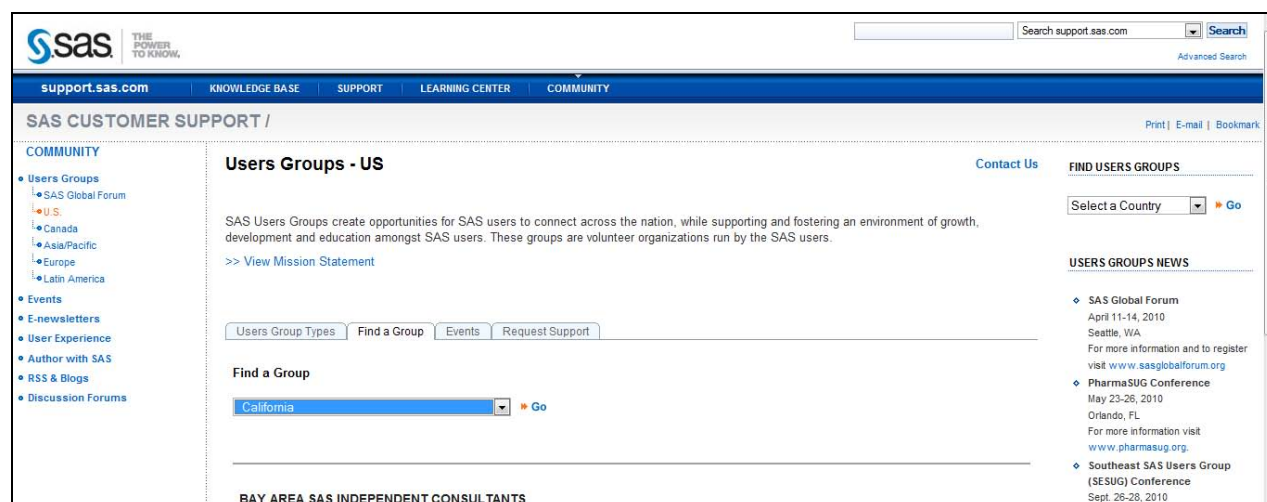
But to get to this point, we'll first have to:

- ✓ Figure out what SAS User Group information we need and where to get it (hint: support.sas.com!)
- ✓ Learn a little bit about Google Maps and how it uses KML/XML feeds
- ✓ See how everything can be tied together and automated with SAS and the Google Maps JavaScript API

Let's begin by figuring out where – and how – to get our data!

GOT DATA?

If you click your way over to support.sas.com, you'll find the good folks at SAS have made plenty of user group information available – name, address, phone, e-mail, website URLs, etc.

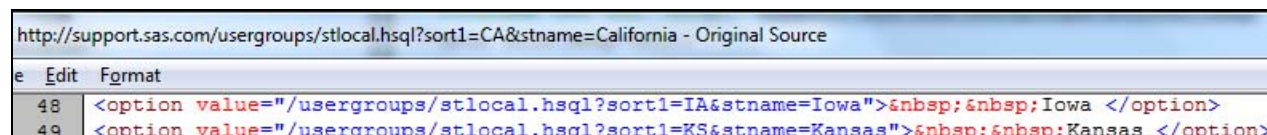


SAS CUSTOMER SUPPORT – U.S. USER GROUPS

That's the good news. The bad news is that the data, at least at first glance, doesn't appear to be readily accessible to a SAS program. Selecting a state from a list box and clicking Go causes semi-structured data for that state to be displayed at the bottom of the page. To view all of the U.S. User Groups, you could wind up repeating this process 50 times.

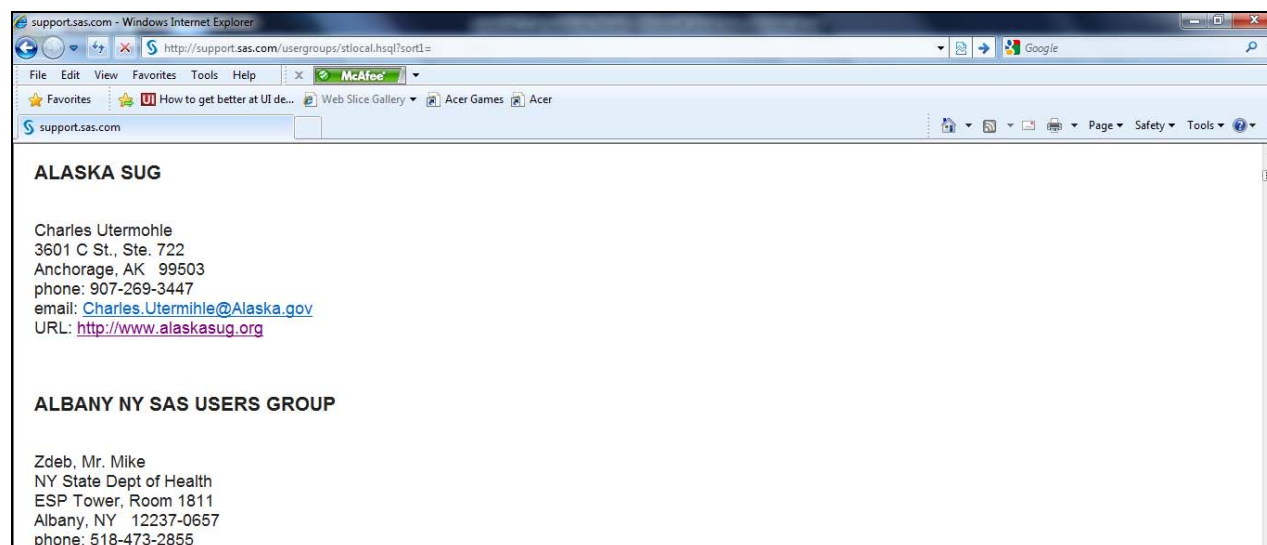
On the other hand, you could do a little snooping and use the screen-scraping capabilities of the SAS URL Access Method to help automate this process!

First, the snooping. Right-click on the web page, select *View Source*, and you'll see that selecting a state and clicking Go causes a query to be issued with the selected state as a parameter:



SAS CUSTOMER SUPPORT – U.S. USER GROUPS (HTML)

Specify nothing for the state parameter – i.e., <http://support.sas.com/usergroups/stlocal.hspl?sort1=> – and you'll be pleasantly surprised to find that what you get back is a screen with all SAS U.S. User Groups:



SAS CUSTOMER SUPPORT – U.S. USER GROUPS

Now we can turn our attention to figuring out how we can use SAS to “scrape” the user group-related fields we want from this web page of semi-structured data.

THE URL-Y CODE GETS THE DATA!

If there's some "reasonable" structure to a web page's underlying HTML, you'll be able to write a SAS program that uses the URL Access Method to read the web page and "scrapes" the data you need. If you right-click on the web page and select *View Source*, you'll see that the SAS User Group entries look like this:

```
<tr><td colspan="2">
<h3>BAY AREA SAS INDEPENDENT CONSULTANTS</h3>
</tr>
<tr><td>
Karp, Mr. Andrew<br />
Sierra Info. Services, Inc.<br />
19229 Sonoma Hwy PMB 264<br />
Sonoma, CA &nbsp; 95476<br />
phone: 707-996-7380<br />
email: <a href="mailto:SierraInfo@aol.com">SierraInfo@aol.com</a><br />
URL: <a target="_blank"
href="http://www.sierrainformation.com/basic.htm">http://www.sierrainformation.com/basic.htm</a>
<br /><br />
</td>
```

SAS USER GROUP ENTRY (HTML)

With this in mind, the following code scrapes the needed fields – user group name, website, and zip code – from the SAS web page, and also picks up latitude, longitude, state name, and state code from the sashelp.zipcode dataset:

```
*==> Grab U.S. User Group info from SAS website;
filename www url 'http://support.sas.com/usergroups/stlocal.hsql?sort1=';
data urls(keep=UserGroup UserGroupURL UserGroupZip);
length UserGroup UserGroupURL UserGroupZip $ 512;
retain UserGroup UserGroupURL UserGroupZip ' ';
infile www url end=eof;
input;
if index(_infile_,"<h3>") then do;
  if UserGroup^=' ' then do;
    output;
    UserGroupZip=' ';
  end;
  UserGroup=upcase(scan(_infile_,2,'<>'));
  UserGroupURL='http://support.sas.com/usergroups/stlocal.hsql?sort1=';
end;
zip=compress(scan(scan(_infile_,-2,"<"),-1," "),'- ');
if (length(zip)=5 or length(zip)=9) and ^verify(zip,'0123456789 ') then
  UserGroupZip=substr(zip,1,5);
if index(_infile_,"URL:") then
  UserGroupURL=lowercase(scan(_infile_,4,''));
if eof and UserGroup^=' ' then
  output;

proc sort data=urls(where=(UserGroupZip^="")) nodupkey;
by UserGroup UserGroupURL UserGroupZip;

*==> Add latitude/longitude from sashelp.zipcode;
libname pc 'c:\sgf2010';
data pc.GroupsAndZips(keep=UserGroup UserGroupURL UserGroupZip
                        lat lng statecode statename);
set urls;
Zip=input(UserGroupZip,5.);
set sashelp.zipcode key=zip;
saverr=_error_;
_error_=0;
if saverr=0;
lat=y; lng=x;

proc sort data=pc.GroupsAndZips nodupkey;
by statename usergroup;
```

SAS "SCREEN SCRAPER" CODE (URL ACCESS METHOD)

The output of this code – a SAS dataset containing data for the map – is shown on the following page (please note that user groups without a zip code or latitude/longitude are dropped – hope I didn't lose one of your favorites!).

UserGroup	UserGroupURL	UserGroupZip	Two-letter abbrev. for state name.	Full name of state/territory	lat	lng
BIRMINGHAM USERS GROUP FOR SAS	http://www.epi.soph.uab.edu/bugs/	35244	AL	Alabama	33.36171	-86.788783
ALASKA SUG	http://www.alaskasug.org	99503	AK	Alaska	61.193435	-149.888062
VALLEY OF THE SUN SAS USERS GP	http://www.valsup.org	85080	AZ	Arizona	33.633614	-112.111004

SAS DATASET OF U.S. USER GROUPS (FOR MAPS)

GOOGLE MAPS KML 101

Now that the hard work is done, let's start map-making! Google Maps – and Microsoft's Bing Maps, for that matter – love XML. In fact, Google created a special “dialect” of XML called KML (**Key**hole **Ma**rku**p** **L**anguage) with geospatial-oriented tags specifically for their Google Earth product. A subset of KML is also understood by Google Maps, and that's what we'll use to construct our first SAS User Group Map. Perhaps the simplest way to explain KML is with an example – shown below is KML for the Alabama SAS User Group:

```
<Folder>
  <name>Alabama</name>
  <open>1</open>
  <Placemark>
    <name>BIRMINGHAM USERS GROUP FOR SAS</name>
    <description>
      <![CDATA[ Website: <a href="http://www.epi.soph.uab.edu/bugs/">http://www.epi.soph.uab.edu/bugs/</a> ]]>
    </description>
    <Point>
      <coordinates>-86.788783,33.36171,0</coordinates>
    </Point>
  </Placemark>
</Folder>
```

SCREENSHOT OF KML FOR ALABAMA SAS USER GROUP

In the above, the **Folder** tag is used to create a collapsible folder containing the user groups for the state specified by the **name** tag. As its name suggests, the **open** tag indicates the folder should be displayed in an open state. Following this, the **Placemark** tag is used to denote the start of the user group specified in the following **name** tag. The **description** tag's being used to provide a link to the user group's website. Finally, the **Point** tag is used to specify information about the user group's location, with the longitude and latitude identified by the **coordinates** tag.

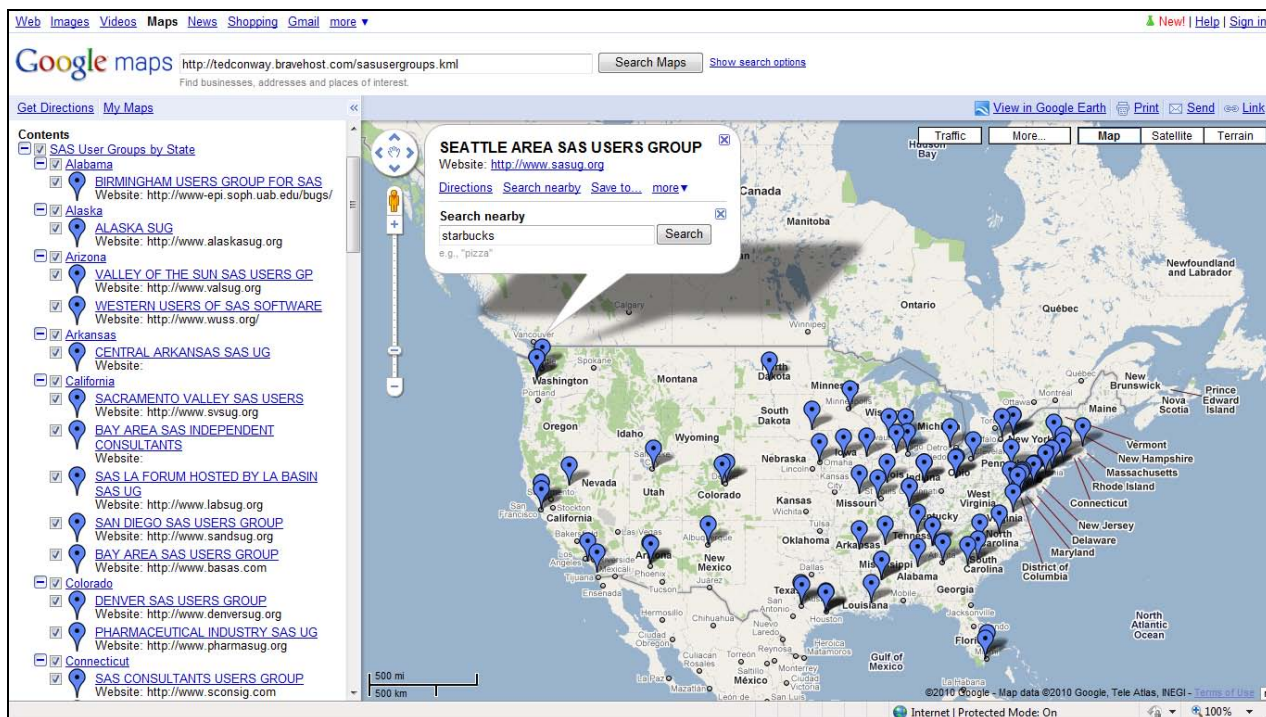
Below is the code used to create the KML for the SAS User Groups – note that the SAS FTP Access Method is being used to write the KML directly to a website, in this case one hosted by the (free) Bravenet web hosting service.

```
*==> 1. Write KML file to website for use with Google and Bing maps
      2. Create a collapsible folder for each state
      3. Include links to user group websites;
filename outftp ftp '/web/tedconway.bravehost.com/sasusergroups.kml'
      host='ftp10.bravehost.com' user='*****' pass='*****';
data _null_;
file OUTFTP lrecl=2000;
set pc.GroupsAndZips end=eof;
by statename;
if _n_=1 then
  put '<?xml version="1.0" encoding="UTF-8"?>' /
      '<kml xmlns="http://www.opengis.net/kml/2.2">' /
      '<Folder>' '<name>SAS User Groups by State</name><open>1</open>';
if first.statename then
  put '<Folder><name>' statename +(-1) '</name><open>1</open>' ;
put '<Placemark>' /
    '<name>' usergroup '</name>' /
    '<description>' /
    '<![CDATA[Website: <a href="' UserGroupURL +(-1) '">' UserGroupURL +(-1) '</a>]]></description>' /
    '<Point><coordinates>' lng +(-1) ',' lat +(-1) ',0</coordinates></Point>' /
    '</Placemark>';
if last.statename then put '</Folder>';
if eof then put '</Folder></kml>';
```

SAS CODE – CREATE KML FOR ALL USER GROUPS

How the generated KML appears when viewed with Google Maps appears on the following page.

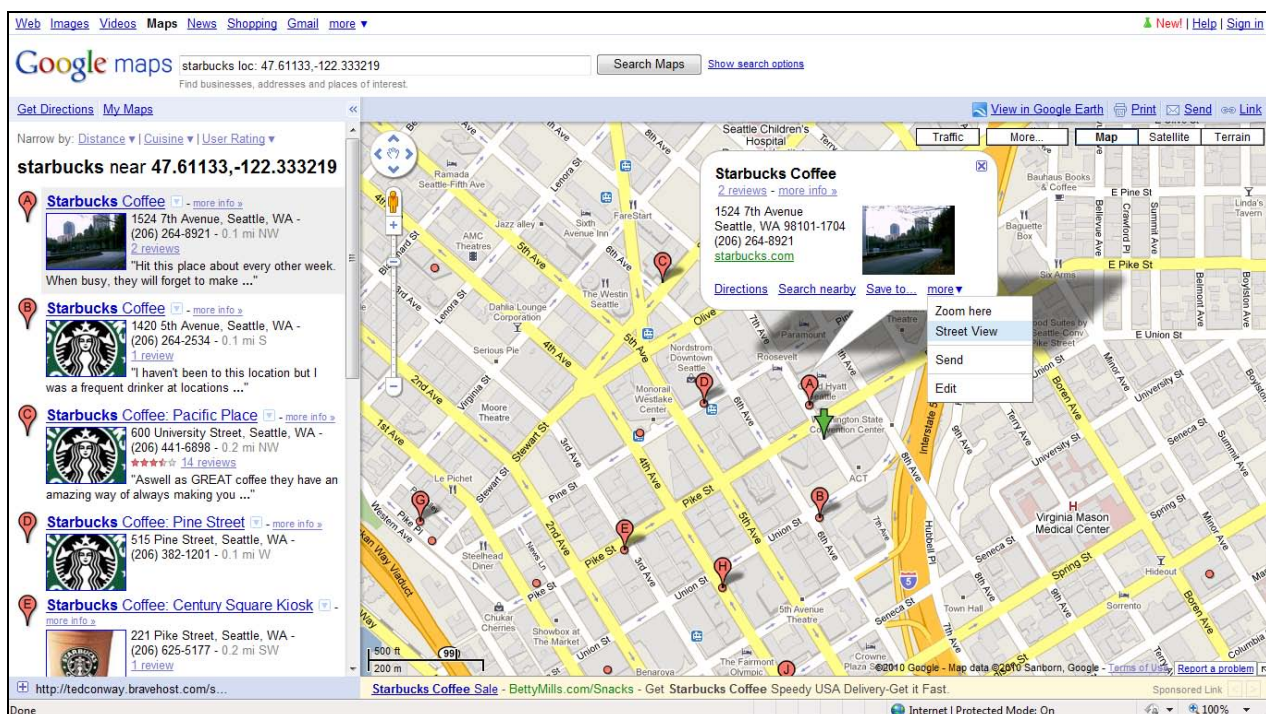
SAS U.S. USER GROUPS – GOOGLE MAPS (KML)



GOOGLE MAPS – KML VERSION

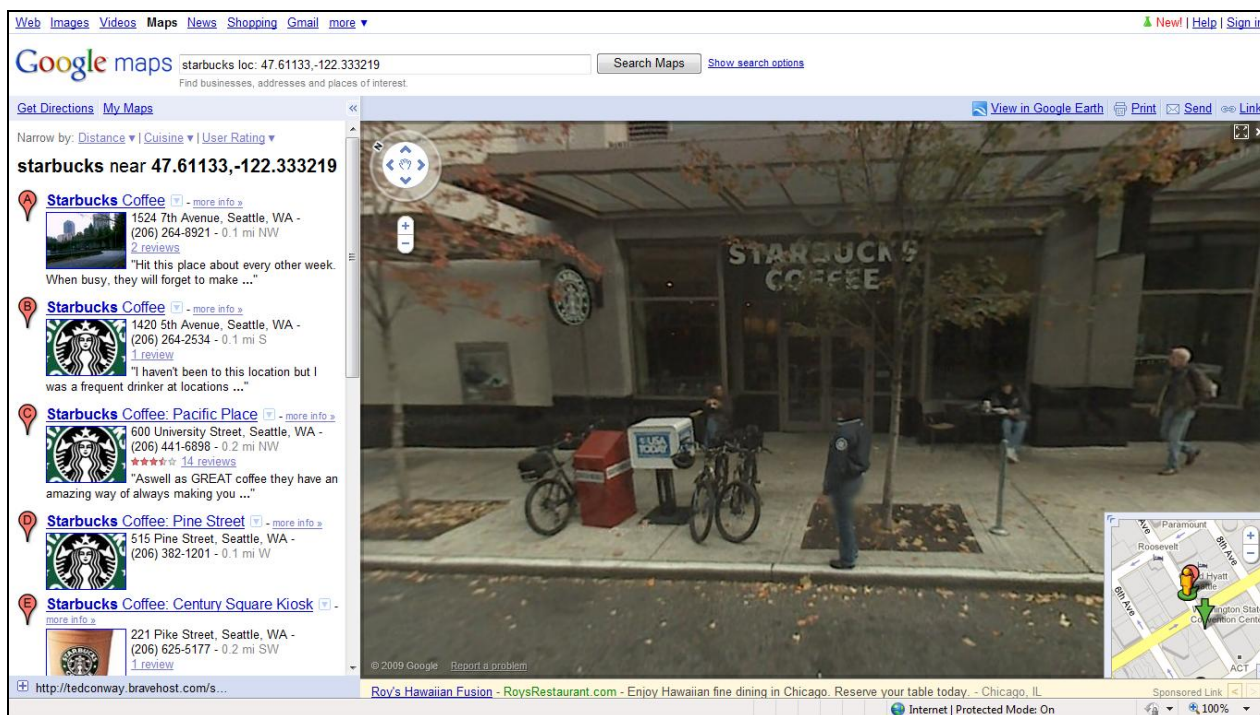
- ✓ The left sidebar contains a collapsible list of the SAS User Groups by state
- ✓ Markers show the location of the SAS User Groups on the map
- ✓ Infowindows pop-up when a user group is selected, either by clicking on its marker or left sidebar entry, and SAS User Group websites are opened in a new window when Infowindow links are clicked

To render KML, you simply specify a URL in the Google Maps search box or a KML file name in the browser address bar, e.g., <http://maps.google.com/maps?q=http://tedconway.bravehost.com/sasusergroups.kml>. All of the Google Maps features you know and love – local search, zoom, street view, Flickr photos, etc. – are available for hours of clicking fun! For example, a search for a nearby Starbucks yields plenty of hits in Seattle:



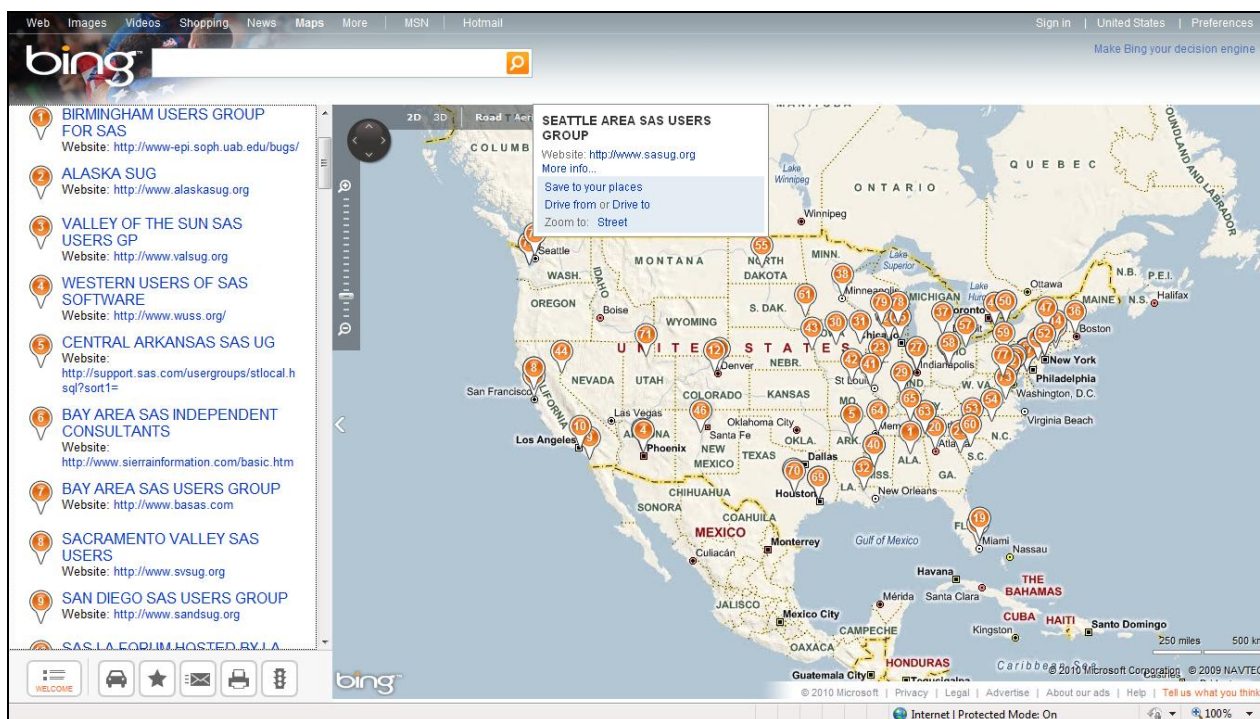
GOOGLE MAPS – LOCAL SEARCH

And selecting Street View for a specific location takes us right to the front door!



GOOGLE MAPS –STREET VIEW

WHAT ABOUT MICROSOFT?

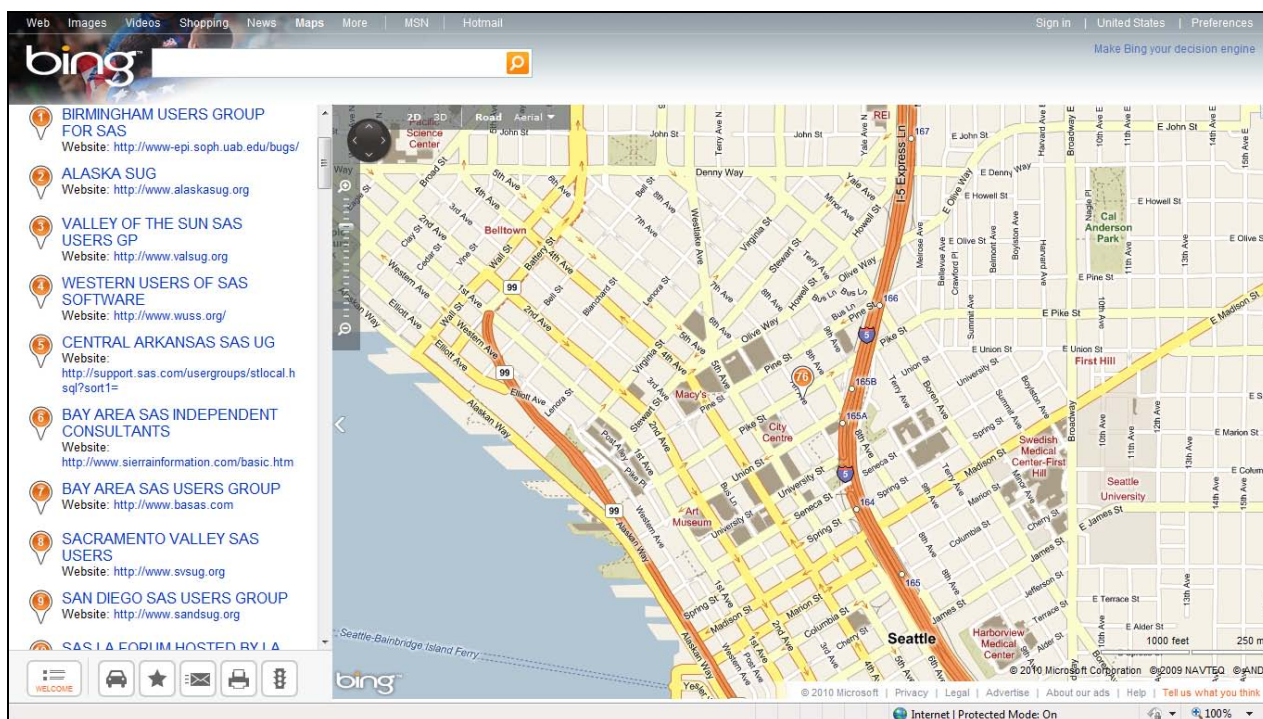


BING MAPS –SAS U.S. USER GROUPS

Since Seattle's hosting SGF this year, it's only fitting to point out that KML— while geared towards Google — can also be rendered by Microsoft Bing Maps. Just specify a KML file name in the browser address bar, e.g., <http://www.bing.com/maps/?mapurl=http://tedconway.bravehost.com/sasusergroups.kml>.

When it comes to rendering KML, both Microsoft and Google have their own “quirks.” Look close, and you’ll see that Bing doesn’t (yet) support KML folders. However, it does provide clickable links to the user group sites in the left sidebar, and managed to display URLs with parameters, both of which Google Maps failed to do. Before leaving the world of KML, we’ll take a quick look at some neat Bing Maps features on the following page.

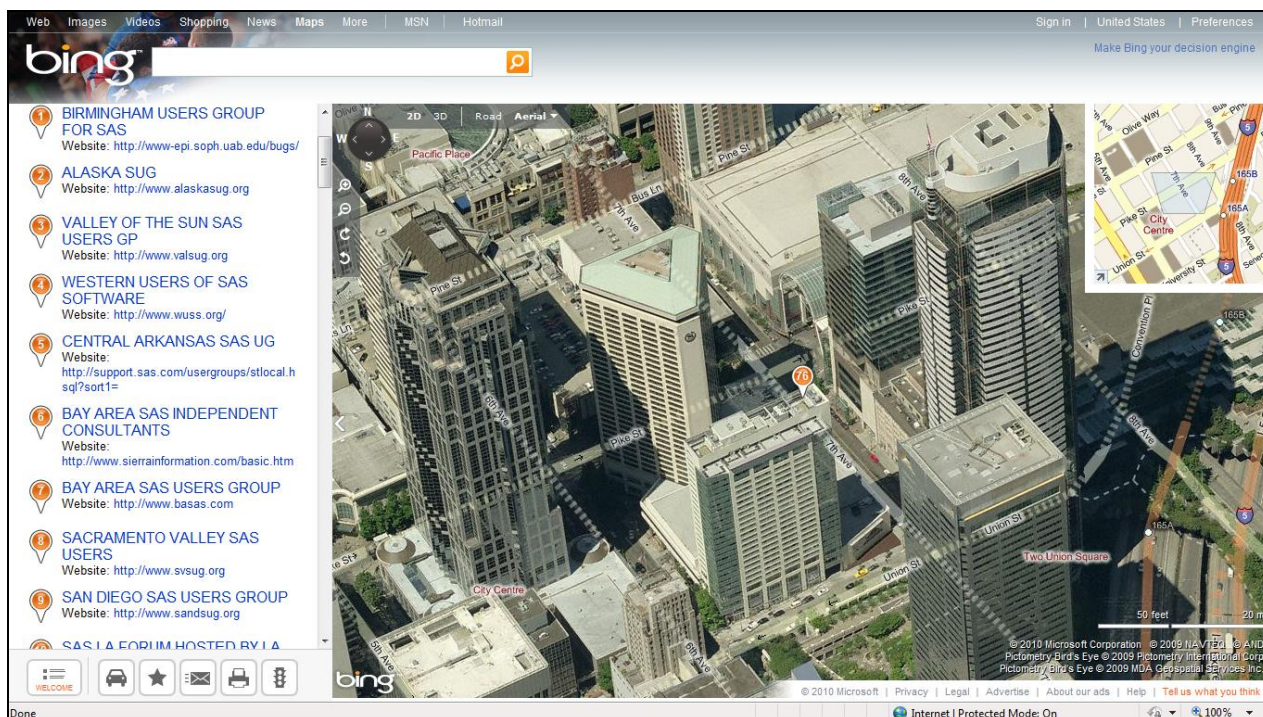
BING MAPS (CONTINUED)



BING MAPS – STREET LEVEL

All of the Bing Maps features are available, so you can explore to your heart's content. For example, zooming down to street level from the SAS Users Groups map results in the display above.

One particularly nifty feature of Bing Maps is "Bird's-Eye View." Click on the *Aerial* drop-down menu, select *Bird's Eye*, and you'll be treated to the panorama below. Nice, eh? Look closely at the upper right corner and you'll spot the Washington State Convention Center!



BING MAPS – BIRD'S EYE VIEW

And that concludes our brief tour of KML-based SAS User Group Maps – be sure to check out the links in the *References* for more information. Now on to a quick how-to on Google API-based maps!

GOOGLE MAPS API 101

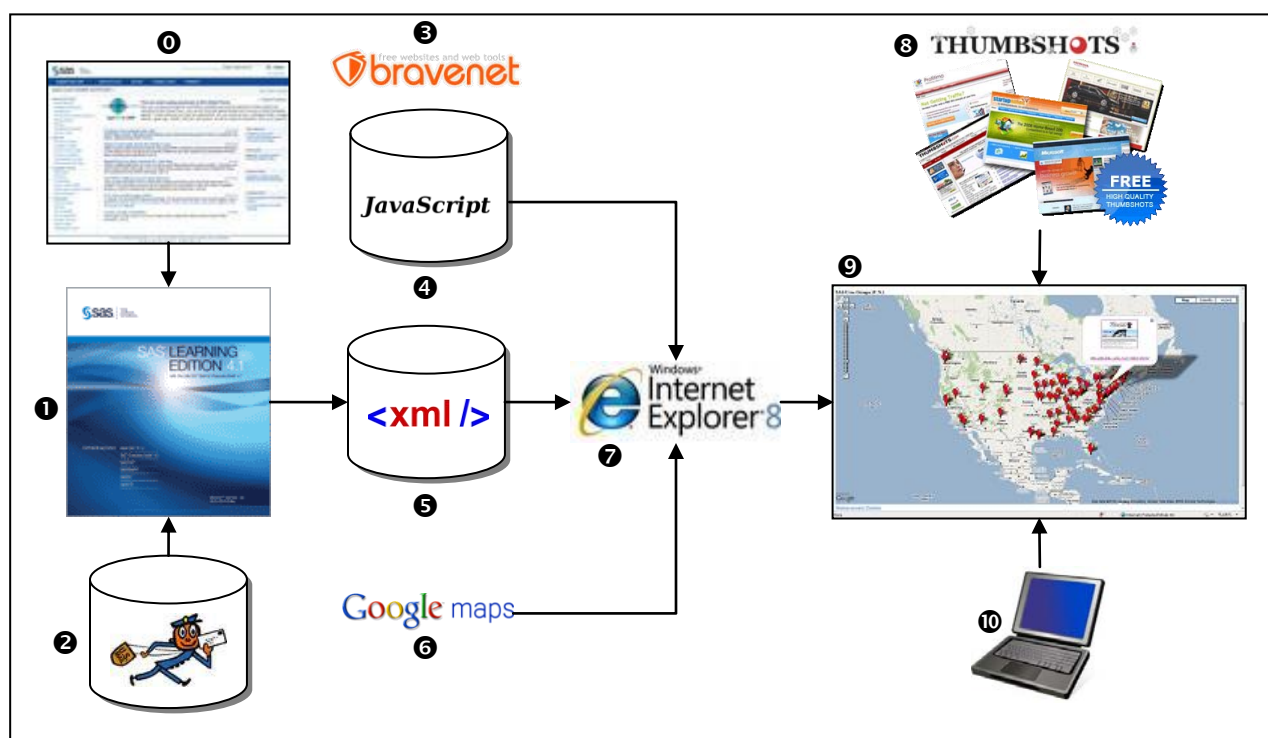
While KML-based maps are appealing from the standpoint that Google Maps does all of the “heavy lifting,” sometimes you’ll want to exert a finer level of control over the appearance and behavior of maps.

And that’s where the Maps API – which lets you embed maps in your own web pages with JavaScript – comes in!

To get started with the Google Maps V2 API you’ll need the following:

- ✓ A web hosting service that supports JavaScript – I used the free web hosting provided by Bravenet.com, which also includes an FTP server (surprisingly, Google Sites does not support JavaScript or FTP!).
- ✓ A Google Maps API key – sign up at <http://code.google.com/apis/maps/signup.html>
- ✓ Access to the Google Maps API Developer's Guide (<http://code.google.com/apis/maps/documentation/index.html>)
- ✓ Access to the Google Maps API Reference (<http://code.google.com/apis/maps/documentation/reference.html>)
- ✓ A free license for Thumbshots website link previews – sign up at <http://www.thumbshots.com/Default.aspx>

JAVASCRIPT API-BASED GOOGLE MAPS (V2)



GOOGLE API MAPS – A HIGH-LEVEL “ROADMAP”

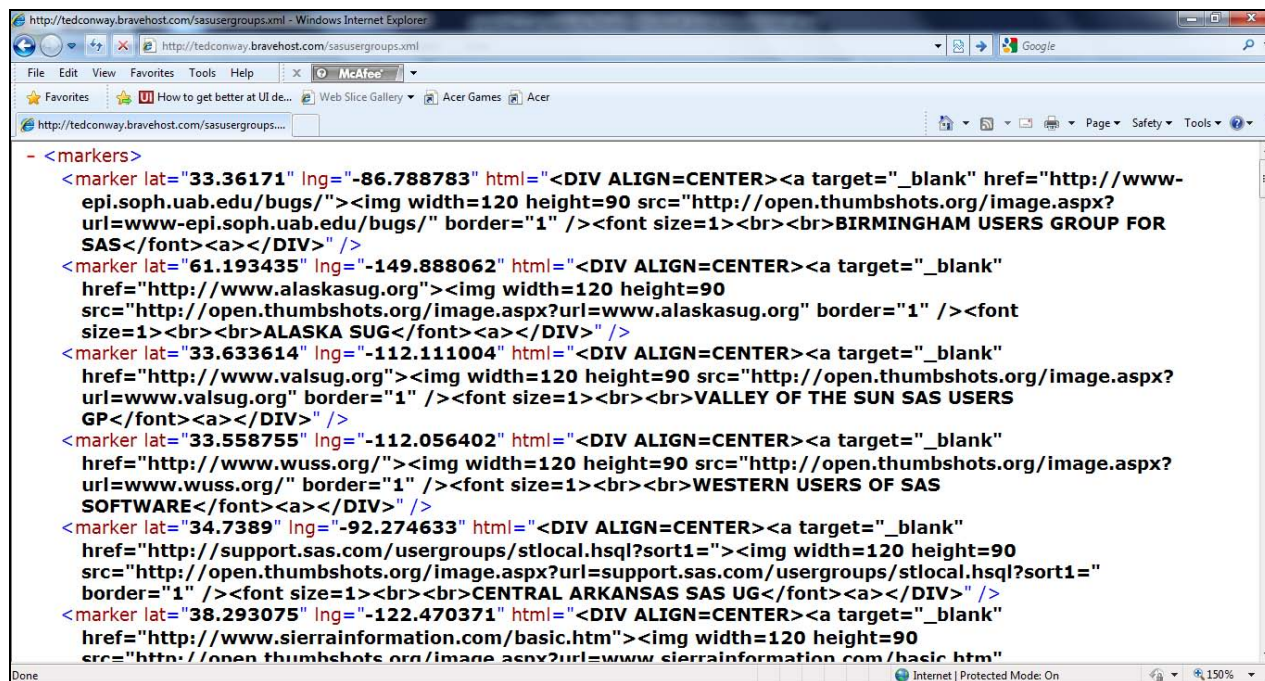
We’ll get to the details in the next couple of pages, but let’s first take a high-level look at all the pieces that have to come together to create the SAS User Groups Map using the Google Maps API.

- ✓ SAS Learning Edition **1** scrapes SAS User Group information – user group name, website URL, zip code – from support.sas.com **0**, merges it with zip code information – latitude, longitude, state name, state code – from sashelp.zipcode **2**, and writes an XML file **5** on the web server **3**. This is essentially the same as the KML processing we saw earlier, although the XML has a different, user-defined format (see below) and includes a little extra HTML to generate a “sneak preview” image of the SAS User Group websites using Thumbshots.com **8**.
- ✓ Internet Explorer **7** then launches your Google Maps JavaScript API code **4**, which reads the XML data **5** and uses the Google Maps **6** and Thumbshots.com **8** web services to generate a customized Google Map **9** that’s viewed on the user’s laptop **10**.

As you can see, a custom Google Map is certainly much more complicated to produce than a custom SAS/GRAPH map, but until SAS makes our lives easier with an ODS GOOGLEMAPS destination (*I’m only half-joking, SAS!*), we’ll have to play with the hand that Google’s dealt us. So, on to the XML and JavaScript requirements!

XML MARKS THE SPOT(S)

In our KML example, we used the pc.GroupsAndZips SAS dataset created on page three. Now, we'll use that same dataset to create an XML file containing the geospatial information needed by the Google Maps API. Again, perhaps the simplest way to explain what we're looking for in terms of XML is with an example – shown below is the XML for the first few user groups:



GOOGLE API – SAS USER GROUPS XML

In the above XML, the **Marker** tag is used to denote the start of a user group. As their names suggest, the **lat** and **lng** attributes are respectively used to specify latitude and longitude. Finally, the **html** tag is used to specify the HTML for the Infowindow that will pop-up when a marker is moused-over or clicked (see Thumbshots.com for a complete description of the link preview HTML requirements). Unlike the KML-based map, our custom API-based map has no sidebar, so no **folders** are specified. Unlike the KML tags, these XML tags are user-defined – the JavaScript code you write to parse the XML (next page) will just have to match the tags you've chosen.

Below is the code used to create the XML for the SAS User Groups – again, the SAS FTP Access Method is being used to write the XML directly to a website, in this case one hosted by the Bravenet web hosting service.

```
*==> 1. Write XML file to website for use with Google Maps API
      2. Include latitude, longitude, HTML for pop-up Infowindow
      3. Include links to user group websites, Thumbshots preview code;
filename outftp ftp '/web/tedconway.bravehost.com/sasusergroups.xml'
      host='ftp10.bravehost.com' user='*****' pass='*****';
data _null_;
file OUTFTP lrecl=2000;
set pc.GroupsAndZips end=eof;
if _n_=1 then put '<markers>';
UserGroupUrl=tranwrd(UserGroupUrl, '&', '&'); /* Replace & with &amp; */
UserGroupUrl2=substr(UserGroupUrl, 8); /* Strip off http:// for Thumbshots */
put '<marker lat="' lat + (-1) '"' lng="' lng + (-1) /* Avoid >, <, ' in HTML */
    '" html="&lt;DIV ALIGN=CENTER&gt;&lt;a target=&quot;_blank&quot; href=&quot;'
    UserGroupURL + (-1) '&quot;&gt;&lt;img width=120 height=90
src=&quot;http://open.thumbshots.org/image.aspx?url='
    UserGroupURL2 + (-1) '&quot; border=&quot;1&quot; /&gt;&lt;font
size=1&gt;&lt;br&gt;&lt;br&gt;'
    UserGroup + (-1) '&lt;/font&gt;&lt;a&gt;&lt;/DIV&gt;' />';
if eof then
put '</markers>';
```

SAS CODE– CREATE XML FOR ALL USER GROUPS

And now, on to the last piece in the puzzle – the HTML and JavaScript API code that will read this XML data and use the API to generate our custom Google Map. The HTML on the next page was cobbled together using code snippets from the Google Maps API V2 site and Mike Williams' API Tutorial website (see *References*).

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>SAS User Groups (U.S.)</title>
<script
src="http://maps.google.com/maps?file=api&v=2&sensor=false&key=ABQIAAAAGQd
OKaU2qiqmrj9f48M97hS_BY295Oavr-F4FWsBnGJtxzu9vRQUi483e7yJyqYTbRa9743JcLJyCQ"
type="text/javascript"></script>
</head>
<script src="http://gmaps-utility-
library.googlecode.com/svn/trunk/mapiconmaker/1.1/src/mapiconmaker.js"
type="text/javascript"></script>
<table align="center">
<body onunload="GUnload()">
<a name="mapstart"><font size=3><b><span>SAS User Groups
(U.S.)</span></b></font></a><br>
<div id="map"></div>
<font size=1><a href="http://www.thumbshots.com" target=" blank" title="Thumbnails
Previews by Thumbshots">Thumbnails powered by Thumbshots</a></font>
</table>
<noscript><b>Sorry, JavaScript required.</b></noscript>
<script type="text/javascript">
document.getElementById("map").style.width =
.98*document.documentElement.clientWidth+'px';
document.getElementById("map").style.height =
.93*document.documentElement.clientHeight+'px';
</script>
<script type="text/javascript">
//
if (GBrowserIsCompatible()) {
var gmarkers = [];

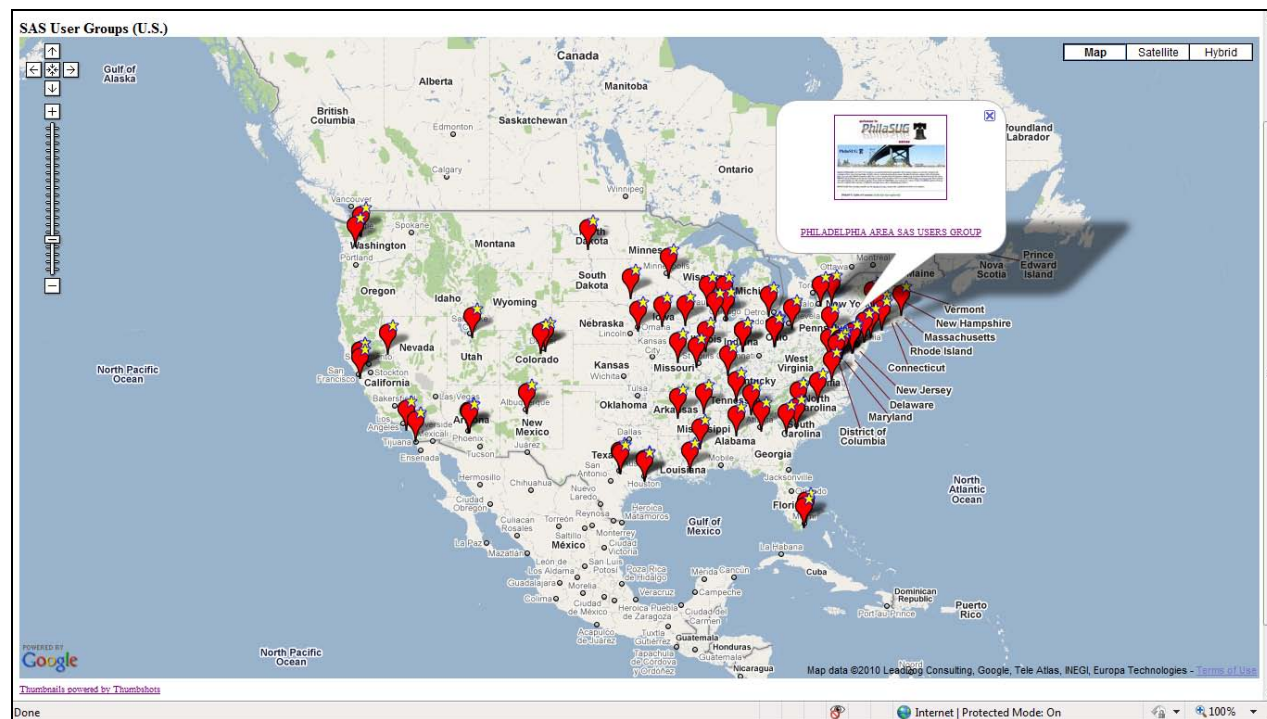
// Function to create the marker and set up the event window
function createMarker(point,html) {
var newIcon = MapIconMaker.createLabeledMarkerIcon({addStar: true, primaryColor:
"#ff0000"});
var marker = new GMarker(point, {icon: newIcon});
GEvent.addListener(marker, "mouseover", function() {marker.openInfoWindowHtml(html);
});
GEvent.addListener(marker, "click", function() {marker.openInfoWindowHtml(html);});
gmarkers.push(marker);
return marker; }

// function picks up the click and opens the corresponding info window
function myclick(i) {GEvent.trigger(gmarkers[i], "click");}

// Create and center the map
var map = new GMap2(document.getElementById("map"));
map.addControl(new GLargeMapControl());
map.addControl(new GMapTypeControl());
map.setCenter(new GLatLng(39.042939, -95.769657), 4);
var mm = new GMarkerManager(map);

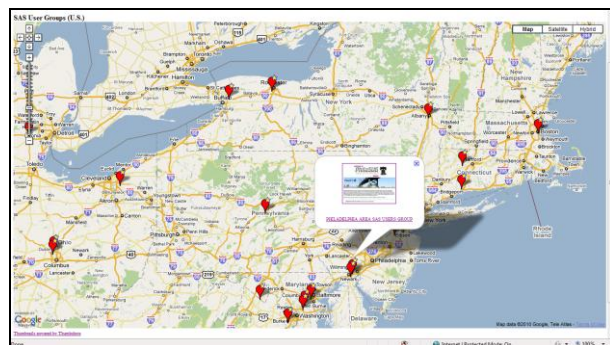
// Read data from the xml file - latitude, longitude, pop-up HTML
GDownloadUrl("http://tedconway.bravehost.com/sasusergroups.xml", function(doc) {
var xmlDoc = GXml.parse(doc);
var markers = xmlDoc.documentElement.getElementsByTagName("marker");
for (var i = 0; i &lt; markers.length; i++) {
var lat = parseFloat(markers[i].getAttribute("lat"));
var lng = parseFloat(markers[i].getAttribute("lng"));
var point = new GLatLng(lat,lng);
var html = markers[i].getAttribute("html");
var marker = createMarker(point,html);
map.addOverlay(marker); }
});}
else
{alert("Sorry, the Google Maps API is not compatible with this browser"); }
//]]&gt;
&lt;/script&gt;&lt;/body&gt;&lt;/html&gt;
</pre>
</div>
<div data-bbox="675 100 859 127" data-label="Text">Specify browser title and your Google Maps API key</div>
<div data-bbox="691 227 866 242" data-label="Text">Define map within a table</div>
<div data-bbox="739 258 857 272" data-label="Text">Specify map title</div>
<div data-bbox="697 319 862 333" data-label="Text">Credit Thumbshots.com</div>
<div data-bbox="729 375 860 388" data-label="Text">Size map for client</div>
<div data-bbox="751 478 862 492" data-label="Text">Create markers</div>
<div data-bbox="575 520 862 534" data-label="Text">Open Infowindows on mouseovers &amp; clicks</div>
<div data-bbox="677 620 860 634" data-label="Text">Open Infowindows on click</div>
<div data-bbox="659 654 868 668" data-label="Text">Create, center, and zoom map</div>
<div data-bbox="718 745 866 798" data-label="Text">Read XML file; parse latitude, longitude for points, and HTML for pop-up Infowindows</div>
<div data-bbox="729 825 867 840" data-label="Text">Add makers to map</div>
<div data-bbox="316 908 890 921" data-label="Text">GOOGLE MAPS JAVASCRIPT API CODE – CUSTOM SAS USER GROUP MAP (RESULTS ON NEXT PAGE)</div>
<div data-bbox="486 937 510 952" data-label="Page-Footer">10</div>
```


SAS U.S. USER GROUPS – GOOGLE MAPS (API)



GOOGLE API – CUSTOM SAS U.S. USER GROUPS MAP

- ✓ Markers show the location of the SAS User Groups on the map
- ✓ Infowindows pop-up with a preview of the user group's website when a marker is clicked on or hovered over
- ✓ Google Maps features such as zoom are still available (see example, below left)
- ✓ Clicking on the preview image or link in the pop-up Infowindow opens the user group's website in a new window (see example, below right)



ZOOM IN FOR GREATER DETAIL



LINK TO PHILLY SAS USER GROUP WEBSITE

Unlike KML maps, API-based maps are viewed at the website on which they're defined – not at maps.google.com.

In our example, the map is hosted at <http://tedconway.bravehost.com/sasusergroups.html#mapstart> (#mapstart is just an anchor link that was inserted to allow us to jump past the banner ads inserted by the ad-supported Bravehost!).

CONCLUSION

Should you need to go beyond SAS/GRAPH and present your data on Google or Bing maps, you'll find that SAS can still help make life much easier. In the examples shown in this paper, we've seen how SAS features can be used to automatically extract and transform raw data on a web page into KML and XML that's rendered into fully-interactive, attractive, web-based maps. And that's just the tip of the iceberg. Start reading the Google and Microsoft Maps documentation (see *References* on the next page) to get some new ideas, crank out some SAS code and macros to implement them, and you'll find that putting yourself on the map is easier than ever!

REFERENCES

There's certainly a wealth of freely available information on the web on the topic of Google and Microsoft Maps. For starters, you might want to take a gander at:

- ✓ **Google Maps API Documentation** (<http://code.google.com/apis/maps/>)
- ✓ **Google KML Documentation** (<http://code.google.com/apis/kml/documentation/>)
- ✓ **Mike Williams' Google Maps API Tutorial** (<http://econym.org.uk/gmap/>)
- ✓ **Microsoft Bing Maps for Developers** (<http://www.microsoft.com/maps/developers/>)

There are also many SAS papers worth checking out, including the following:

- ✓ **Notes from an Intersection: Google Earth @ SAS** (<http://support.sas.com/resources/papers/proceedings09/233-2009.pdf>), by Carol Martell, is ostensibly geared towards Google Earth, but you'll find it's equally useful for Google Maps development, and there's even a SAS XML lesson thrown in for good measure.
- ✓ **Put Your Customers on the Map: Integrating SAS/GRAPH and Google Earth** (<http://www2.sas.com/proceedings/forum2008/252-2008.pdf>), by Daniël Kuiper and Koen Vyverman, is also Google Earth-centric, but it'll each teach you more than a trick or two about KML, XML, and Google Maps, as it nimbly jumps back and forth between the worlds of SAS/GRAPH and Google Earth.
- ✓ **Using SAS and Google Earth to Access and Display Air Pollution Data** (<http://www2.sas.com/proceedings/forum2008/253-2008.pdf>), by Josh Drukenbrod and David Mintz, is another Google Earth-oriented paper, but it contains a nice treatment of KML that Google Maps users will also find helpful
- ✓ **Cheap Geocoding: SAS/GIS and Free TIGER Data** (<http://support.sas.com/rnd/papers/sugi30/CheapGeocoding.pdf>), by Ed Odom and Darrell Massengill, helped get me going with its nice overview of centroid-level geocoding and some of the associated issues (e.g., the problem of identically-located centroids).
- ✓ **The URL-y Show: Using SAS LE and the URL Access Method to Retrieve Stock Quotes** (<http://www2.sas.com/proceedings/sugi28/073-28.pdf>), by yours truly, was an early experiment of mine that used the SAS URL Access Method to "scrape" stock quotes from the Web.

What's the best JavaScript tutorial? is tackled at <http://stackoverflow.com/questions/646032/whats-the-best-javascript-tutorial>.

Finally, if you want to see the ghost-of-interactive-maps-future, check out the dazzling **TED2010 Augmented-Reality Maps Demo** given by Microsoft's Blaise Aguera y Arcas at http://www.ted.com/talks/blaise_aguera.html.

CONTACT INFORMATION

Ted Conway resides in Chicago, Illinois. Spam filters notwithstanding, he can be reached at tedconway@aol.com.

TRADEMARKS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.