Geographic and News APIs: Tips, Code Samples, Explanations, and Downloads

Jason A. Clark
Head of Digital Access and Web Services
Montana State University Libraries
jaclark@montana.edu
twitter.com/jaclark

Tips – Getting Started with Web Services

- Play in the sandbox – pick a service, study it
- Yahoo Query Language: http://developer.yahoo.com/yql/
- Yahoo Developer Central: http://developer.yahoo.com/
- Amazon Web Services Developer Connection: http://developer.amazonwebservices.com/connect/
- Google Code: http://code.google.com/

Tips – Consuming Web Services

- Pick a language or parsing tool
- Find a few data sources (APIs) worth learning about
- Make some requests and look at code in your browser
- Think about added value, more efficient workflows
- Browse around – many language libraries are already written

Tips – Building Web Services

- URIs are your friends – that's your interface
- Use simple CRUD (Create, Read, Update, Delete) functions over HTTP (Get, Delete, Put, Post)
- Keep verbs in API protocol intuitive and memorable
- Start small – simple, read-only requests
- Roll it out, beta version – once it's public you are restricted

Resources and Tools

- New York Times API Tool http://prototype.nytimes.com/gst/apitool/index.html
- Location Aware (blog) http://www.locationawhere.com/
- Where 2.0 Conference http://where2conf.com/where2010
- Geolocation API - diveintohtml5 http://diveintohtml5.org/geolocation.html
- Google Code Playground http://code.google.com/apis/ajax/playground/
- Yahoo Query Language Console http://developer.yahoo.com/yql/console/
- Google Reverse Geocoder - Enter location, get latitude and longitude
    - http://gmaps-samples-v3.googlecode.com/svn/trunk/geocoder/getlatlng.html

Web Services – Building Blocks

1. REQUEST – learn the protocol, ask for the data
2. RESPONSE – receive the data
3. PARSE – pick the pieces you need
4. DISPLAY – format those pieces for display

Geographic and News Web Services – Sample Apps

- Google Ajax Search API - Federate search of Google Data
- Foursquare and Google Maps KML
- Google Static Maps API
- Google Maps API - Local Libraries (XML)
- Google Streetview Widget
- HTML5 Geolocation
- Journal Table of Contents using Google Feed API
- Mobile News Feed using Google Feed API
- New Yorker Fiction Podcast Mashup
- NY Times Bestsellers List

View samples and download code at http://www.lib.montana.edu/~jason/files.php

Code Sample #1: Google Ajax Search API – Javascript and CSS

HTML source:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link href="http://www.google.com/uds/css/gsearch.css" type="text/css"
rel="stylesheet"/>
<style type="text/css">
body {background-color:white;color:black;font-family:Arial,sans-
serif;font-size:small;margin:15px;}
.gsc-control {width:400px;}
</style>
<script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0"
type="text/javascript"></script>
…
```

Javascript source:
(http://www.google.com/uds/api?file=uds.js&v=1.0)

```
…
if (window['google'] != undefined && window['google']['loader'] !=
undefined) {
if (!window['google']['search']) {
window['google']['search'] = {};
google.search.CurrentLocale = 'en';
google.search.ShortDatePattern = 'MDY';
google.search.Version = '1.0';
google.search.NoOldNames = false;
google.search.JSHash = 'b2cf21b87d5348acb0a314b08588b757';
google.loader.ApiKey = 'notsupplied';
google.loader.KeyVerified = true;
google.loader.LoadFailure = false;
}
google.loader.writeLoadTag("script", google.loader.ServiceBase +
"/api/search/1.0/en/b2cf21b87d5348acb0a314b08588b757/default.I.js",
false);
```

xHTML and Javascript Explanation:

- Javascript written by Google – heavy lifting: http://www.google.com/uds/api?file=uds.js&v=1.0
- "google.loader.writeLoadTag" – tells API to run, sets possibilities for search API
- CSS written by Google – formatting and display: http://www.google.com/uds/css/gsearch.css
- Understand these files, but you probably want to leave them as is – code library

Web page for user interface and display

```
…
<script type="text/javascript">
//<![CDATA[
function OnLoad() {
// Create a search control
var searchControl = new GSearchControl();
// Add in a full set of searchers
var localSearch = new GlocalSearch();
searchControl.addSearcher(localSearch);
searchControl.addSearcher(new GwebSearch());
searchControl.addSearcher(new GvideoSearch());
searchControl.addSearcher(new GblogSearch());
searchControl.addSearcher(new GnewsSearch());
searchControl.addSearcher(new GimageSearch());
searchControl.addSearcher(new GbookSearch());
// Set the Local Search center point
localSearch.setCenterPoint("Bozeman, MT");
// tell the searcher to draw itself and tell it where to attach
searchControl.draw(document.getElementById("searchcontrol"));
// execute an inital search
searchControl.execute("library books");
}
GSearch.setOnLoadCallback(OnLoad);
//]]
</script>
<div id="searchcontrol">Loading</div>
```

Web page for user interface and display explanation

- xHTML and javascript that gives action to our script
- Create the interface (GUI) controls with "var searchControl = new GSearchControl();"
- Set the local search parameter with "localSearch.setCenterPoint"
- Set the initial query with "searchControl.execute"
- Decide which pieces of Google data to federate with "searchControl.addSearcher"
- will be populated with script messages OR generated xHTML tags received via our Ajax requests.

Code Sample #2: Foursquare and Google KML

1. Visit your foursquare feeds page. Right click the KML link and copy it to your clipboard (don't download it).
2. Visit Google Maps and paste the link you copied into the search box. Hit enter.
3. Whoops... There is no step 3.

Your foursquare checkins will be highlighted with pins on the map. You can share the map with a friend by clicking either "Send" or "Link" at the top of the map.

Generating KML from Google Spreadsheet
http://earth.google.com/intl/en_uk/outreach/tutorial_spreadsheet.html

Code Sample #3: Google Static Maps API

HTML Markup:

```
<img
src="http://maps.google.com/maps/api/staticmap?center=Bozeman,MT
&zoom=13&size=310×310&markers=color:blue|45.666671,-
111.04859&mobile=true&sensor=false" />
```

Explanation:
It is a simple <img> tag with a call linking into the Google Static Maps API. Included among the customizations:

1. A central location for the map
2. A zoom level for the map
3. A size for our map image
4. A marker color and position using latitude and longitude
5. Set the map to display for mobile settings without a sensor

The necessary pieces to enter are a central location and your specific latitude and longitude. Enter your hometown for the central location in the format of {city, state or principality}. To get your latitude and longitude, visit gmaps-samples-v3.googlecode.com/svn/trunk/geocoder/getlatlng.html and type in your library address. Enter the returned position values as a comma separated string {latitude,longitude}. Here's the image src= markup with cues for where to add your values:

http://maps.google.com/maps/api/staticmap?center={YOURHOMETOWN}&zoom=13&size=310×310&markers=color:blue|{YOUR LATITUDE/LONGITUDE}&mobile=true&sensor=false

Code Sample #4: Google Maps API

Local Libraries (XML) - HTML Markup

```
<div id="map" style="width:75%; height:500px"></div>
```

HTML Explanation:

- Create HTML container where Javascript will load dynamic content
- Add some inline CSS styles to control size of map object

XML File

```
<markers>
 <marker name="St. Johns Co Public Library" address="1960 North Ponce
de Leon Blvd. St. Augustine, Florida 32084"
url="http://www.sjcpls.org/" lat="29.1869" lng="-82.1372" />
 <marker name="Branford Public Library" address="703 Suwannee Ave NW,
Branford, FL 32008-3279" url="http://www.neflin.org/srrl/"
lat="29.963245" lng="-82.93090" />
...
</markers>
```

XML Explanation

- Create XML file with metadata and geographic coordinates; file is named "markers.xml".
- Javascript will use this XML file as instructions for mapping locations

Javascript

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
//<![CDATA[
function load() {
  var map = new google.maps.Map(document.getElementById("map"), {
center: new google.maps.LatLng(29.1869, -82.1372),
zoom: 7,
mapTypeId: google.maps.MapTypeId.TERRAIN,
mapTypeControl: true,
mapTypeControlOptions: {style:
google.maps.MapTypeControlStyle.DROPDOWN_MENU},
navigationControl: true,
navigationControlOptions: {style:
google.maps.NavigationControlStyle.SMALL}
  });
  var infoWindow = new google.maps.InfoWindow;
  // Change this depending on the name of your xml file
  downloadUrl("markers.xml", function(data) {
var xml = data.responseXML;
var markers = xml.documentElement.getElementsByTagName("marker");
```

```
for (var i = 0; i < markers.length; i++) {
 var name = markers[i].getAttribute("name");
 var address = markers[i].getAttribute("address");
 var url = markers[i].getAttribute("url");
 var point = new google.maps.LatLng(
 parseFloat(markers[i].getAttribute("lat")),
 parseFloat(markers[i].getAttribute("lng")));
 var html = '<p style="height:50px;"><strong><a
href="'+url+'">'+name+'</a></strong><br/>'+address+'</p>';
 var marker = new google.maps.Marker({
map: map,
position: point,
 });
 bindInfoWindow(marker, map, infoWindow, html);
}
  });
}
function bindInfoWindow(marker, map, infoWindow, html) {
  google.maps.event.addListener(marker, 'click', function() {
infoWindow.setContent(html);
infoWindow.open(map, marker);
  });
}
function downloadUrl(url, callback) {
  var request = window.ActiveXObject ?
 new ActiveXObject('Microsoft.XMLHTTP') :
 new XMLHttpRequest;
  request.onreadystatechange = function() {
if (request.readyState == 4) {
 request.onreadystatechange = doNothing;
 callback(request, request.status);
}
  };
  request.open('GET', url, true);
  request.send(null);
}
function doNothing() {}
//]]>
</script>
```

Javascript Explanation

- Bring in Google Maps API
- Create Google Maps Object
- Download XML file ("markers.xml"); Parse values to create markers
- Create information windows for data from XML file - These are the Google Maps "pop-up" windows

Code Sample #5: Google Streetview Widget

HTML Markup and CSS Styles:

```
<div id="map_canvas"></div>
<div id="pano"></div>

div#map_canvas {float:left;width:500px;height:300px}
div#pano {float:right;width:500px;height:300px}
```

Explanation:

- Provides a container for map and street view
- Floats map and street view next to each other; specifies width and height

Javascript:

```
<body onload="initialize()" onunload="GUnload()">
...
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=fal
se&amp;key=ABQIAAAApHuYzys7tuBrDzMi_n4tnBQQaDDeMmGffmf_wt4gLMZRq
Rvn0BQaBPW6nqyBUaGp_KbhMX1WCo2-zA"
type="text/javascript"></script>
<script type="text/javascript">
function initialize() {
  if (GBrowserIsCompatible()) {
     var myPano = new
GStreetviewPanorama(document.getElementById("pano"));
     GEvent.addListener(myPano, "error", handleNoFlash);
     var map = new GMap2(document.getElementById("map_canvas"));
     map.setCenter(new GLatLng(<?php echo $lat; ?>,<?php echo
$lng; ?>), 14);
     svOverlay = new GStreetviewOverlay();
     map.addOverlay(svOverlay);
     GEvent.addListener(map,"click", function(overlay,latlng) {
       myPano.setLocationAndPOV(latlng);
     });
   }
}

function handleNoFlash(errorCode) {
  if (errorCode == FLASH_UNAVAILABLE) {
     alert("Error: Flash doesn't appear to be supported by your
browser");
     return;
  }
```

```
}
</script>
```

Explanation:

- Once <body> is loaded the script initializes
- Retrieves lat/lng `map.setCenter(new GLatLng(<?php echo $lat; ?>,<?php echo $lng; ?>), 14);`
- Creates new Street View panorama `GStreetviewPanorama`
- Finds html to display map within `getElementById("map_canvas")`

Code Sample #6: HTML5 Geolocation

HTML Markup and CSS Styles:

```
<section id="wrapper">
...
    <article>
      <p><span id="status">Please wait whilst we try to locate
you...</span></p>
    </article>
</section>
```

Explanation:

- HTML5 tags to hold map data
- Messages from script are written into `<span id="status">`

Javascript:

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=true"></script>
<script>
function success(position) {
  var s = document.querySelector('#status');

  if (s.className == 'success') {
    return;
  }

  s.innerHTML = "found you!";
  s.className = 'success';

  var mapcanvas = document.createElement('div');
  mapcanvas.id = 'mapcanvas';
  mapcanvas.style.height = '400px';
  mapcanvas.style.width = '560px';
```

```
  document.querySelector('article').appendChild(mapcanvas);

  var latlng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
  var myOptions = {
    zoom: 15,
    center: latlng,
    mapTypeControl: false,
    navigationControlOptions: {style:
google.maps.NavigationControlStyle.SMALL},
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new
google.maps.Map(document.getElementById("mapcanvas"),
myOptions);

  var marker = new google.maps.Marker({
      position: latlng,
      map: map,
      title:"You are here!"
  });
}

function error(msg) {
  var s = document.querySelector('#status');
  s.innerHTML = typeof msg == 'string' ? msg : "failed";
  s.className = 'fail';

  // console.log(arguments);
}

if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(success, error);
} else {
  error('not supported');
}

</script>
```

Explanation:

- Checks for W3C geolocation API `if(navigator.geolocation)`
- If successful passes lat/lng to Google Maps API `google.maps.LatLng`
- Writes out map and places a marker `google.maps.Marker`

Code Sample #7: Journal Table of Contents using Google Feed API

HTML and CSS:

```
<style type="text/css">
@import
url("http://www.google.com/uds/solutions/dynamicfeed/gfdynamicfe
edcontrol.css");
</style>
...
<div id="feed-control">
<span style="color:#676767;font-
size:11px;margin:10px;padding:4px;">Loading...</span>
</div>
```

Explanation:

- HTML container that will display TOC list and links
- Simple styles for display and layout

Javascript:

```
 <script type="text/javascript">
    function LoadDynamicFeedControl() {
      var feeds = [
     {title: 'Library Hi-Tech',
      url:
'http://www.emeraldinsight.com/info/RSSFeeds/InsightFeeds/lht_rs
s.xml'
     },
     {title: 'Reference Services',
      url:
'http://www.emeraldinsight.com/info/RSSFeeds/InsightFeeds/rsr_rs
s.xml'
     },
     {title: 'OCLC Systems and Services',
      url:
'http://www.emeraldinsight.com/info/RSSFeeds/InsightFeeds/oclc_r
ss.xml'
     }];
      var options = {
        stacked : true,
        horizontal : false,
        title : "Library Science Journals"
      }

      new GFdynamicFeedControl(feeds, 'feed-control', options);
    }
    // Load the feeds API and set the onload callback.
```

```
    google.load('feeds', '1');
    google.setOnLoadCallback(LoadDynamicFeedControl);
  </script>
```

Explanation:

- Loads different feed values `var feeds`
- Sets up feed controls and defaults `var options`
- Identifies div to use when loading data `GFdynamicFeedControl(feeds, 'feed-control', options);`
- Loads the feeds API `google.load('feeds', '1');`

Final Thoughts

- Start with simpler data formats – RSS and ATOM are well-supported
- Keep experimenting and learning with a single web service, become a seasoned veteran
- Remember the primary actions for using web services: request, response, parse, display

  * Translate these actions into your favorite tool or scripting language