Case Study: Geospatial location analysis & reporting application using SAS, Google Maps & MS Excel

Adney Fernandes, Cognizant Technology Solutions Corporation, Pune, India.

Abstract

This paper discusses a way to integrate Google map using web browser ActiveX control within an Excel workbook to create a practical geospatial reporting application for displaying targets. The geospatial information is obtained by retrieving geo-coordinates (latitude & longitude) information for any US postal address using the Google map API web services, SAS, and Java. Then SAS is used to create state-wise xml files containing target geo-locations. HTML and JavaScript are used to display Google map using Google map Javascript API v3 to load the map into excel and render state-wise target location with contact information such as address and telephone number. This paper then discusses the interaction between excel and web browser component for updating target information using JavaScript in excel using VBA, distance calculation between targets, and also rendering the path between the two points on the map.

INTRODUCTION

Many organizations are planning to use Google maps in their reporting application because of its popularity and the services provided for their maps. Some of the services include geocoding of target addresses, displaying target locations on a Google map, zooming in/out a Google map for more geographical details, directional path rendering between target locations (markers) etc. Since Excel is one of the softwares used for creating reports for business analysis, we need a way to integrate Google maps in Excel so that the Google maps functionality can be leveraged in our reporting and provide better more user friendly reports.

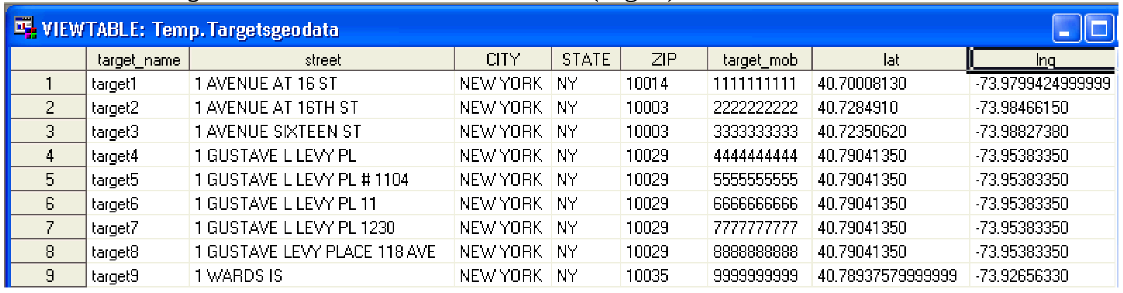
OBTAINING GEOSPATIAL INFORMATION

To show the location of targets on a google map, we first need to obtain geospatial information (latitude and longitude) from the web service provided by Google and then create xml files for our targets.

Retrieve geo-coordinates (latitude & longitude) information for any US postal address using the google map API web services, SAS & Java.

SAS can be used in conjunction with Java to get geo-coordinates using the google maps web services API provided by Google and load the geo-coordinates in an SAS dataset.
The SAS code given below in Appendix A does the following:
(1) for each observation, it instantiates a new java object and passes the target address to the constructor of the java object j1 created.
(2) Calls Java object methods *GetLatitude()* and *GetLongitude()* to get geospatial information and store lat and lng data in the output dataset and then destroys the object. The code is shown in Appendix A.

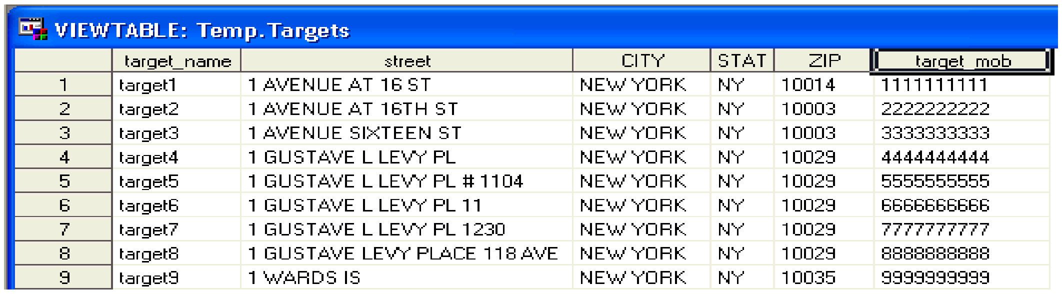
The final dataset *targetsGeoData* will have columns like *target_name*, *street*, *city*, *state*, *zip*, *target_mob*, *latitude* and *longitude* as shown in the screenshot (Fig. 1).


VIEWTABLE: Temp.Targetsgeodata

	target_name	street	CITY	STATE	ZIP	target_mob	lat	lng
1	target1	1 AVENUE AT 16 ST	NEW YORK	NY	10014	1111111111	40.70008130	-73.9799424999999
2	target2	1 AVENUE AT 16TH ST	NEW YORK	NY	10003	2222222222	40.7284910	-73.98466150
3	target3	1 AVENUE SIXTEEN ST	NEW YORK	NY	10003	3333333333	40.72350620	-73.98827380
4	target4	1 GUSTAVE L LEVY PL	NEW YORK	NY	10029	4444444444	40.79041350	-73.95383350
5	target5	1 GUSTAVE L LEVY PL # 1104	NEW YORK	NY	10029	5555555555	40.79041350	-73.95383350
6	target6	1 GUSTAVE L LEVY PL 11	NEW YORK	NY	10029	6666666666	40.79041350	-73.95383350
7	target7	1 GUSTAVE L LEVY PL 1230	NEW YORK	NY	10029	7777777777	40.79041350	-73.95383350
8	target8	1 GUSTAVE LEVY PLACE 118 AVE	NEW YORK	NY	10029	8888888888	40.79041350	-73.95383350
9	target9	1 WARDS IS	NEW YORK	NY	10035	9999999999	40.78937579999999	-73.92656330

Figure 1. Final Dataset

The targets dataset in the temp library contains columns like target_name, street, city, state, and zip as shown in the screenshot (Fig. 2).



	target_name	street	CITY	STAT	ZIP	target_mob
1	target1	1 AVENUE AT 16 ST	NEW YORK	NY	10014	1111111111
2	target2	1 AVENUE AT 16TH ST	NEW YORK	NY	10003	2222222222
3	target3	1 AVENUE SIXTEEN ST	NEW YORK	NY	10003	3333333333
4	target4	1 GUSTAVE L LEVY PL	NEW YORK	NY	10029	4444444444
5	target5	1 GUSTAVE L LEVY PL # 1104	NEW YORK	NY	10029	5555555555
6	target6	1 GUSTAVE L LEVY PL 11	NEW YORK	NY	10029	6666666666
7	target7	1 GUSTAVE L LEVY PL 1230	NEW YORK	NY	10029	7777777777
8	target8	1 GUSTAVE LEVY PLACE 118 AVE	NEW YORK	NY	10029	8888888888
9	target9	1 WARDS IS	NEW YORK	NY	10035	9999999999

Figure 2. Targets Dataset
Java code shown in Appendix B does the following:
(1) Sends the target address for geocoding using the google maps web services api and receives the response in the form of a normal text file either in JSON or XML format.
(2) Parses the response received and if the status is “OK” then extracts the latitude and longitude values from the response and stores in the latitude and longitude class properties respectively.
More information regarding geocoding requests, response and status of requests is provided by Google in the documentation [1].

The java program is compiled and the project is exported to a .jar file (GeoLocation.jar).
Before executing the SAS program the JREOPTIONS parameter(*-Dsas.app.class.path)* file in the *sasv9.cfg* file needs to be set to the location of the exported jar file so that the SAS program can locate it, containing the class “GeoLoc” as shown below:

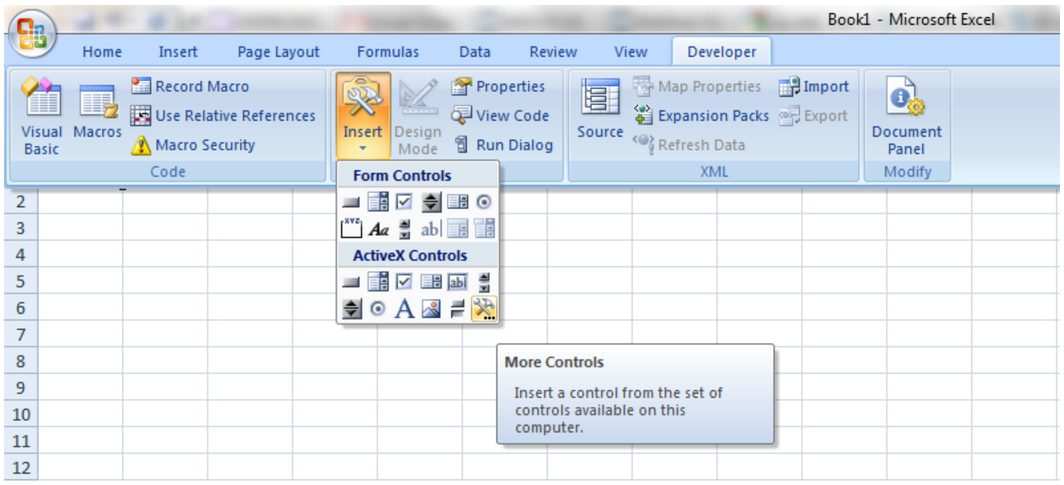
```
-JREOPTIONS=(-
Dsas.app.class.path=C:\PROGRA~1\SAS922~1.EN\SASVER~1\9.2\eclipse\plugins\GeoLocation.j
ar;C:\PROGRA~1\SAS922~1.EN\SASVER~1\9.2\eclipse\plugins\gson-1.4.jar)
```

Using SAS to create state-wise xml files containing target geo-locations.
Once we have the targets geo-locations in a SAS dataset, we now have to create XML files so that we can show the targets (markers) state–wise using Google map with the help of html and javascript.
The SAS code shown in Appendix C is used for creating the state-wise XML files.

The SAS code creates a XML file for each state (eg. KS.xml, TX.xml, NJ.xml etc.) and writes all the targets information to the XML file in which the target resides.

Writing HTML and JavaScript code to display google map

Using Google map JavaScript API v3 to load Google map into excel:
This sub section shows how to integrate a Google map in MS Excel and this is done by adding a web browser ActiveX component in MS Excel.
Given below are the steps to do the same:
(1) Open a new excel sheet and click on the Developer menu -> click insert dropdown -> click more controls and “More control” dialog box will appear as shown in Fig. 3.

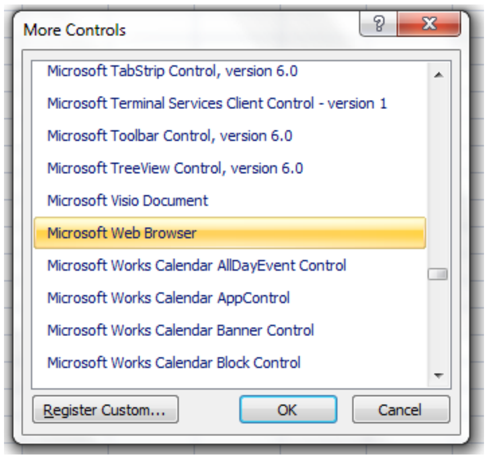


More Controls

Insert a control from the set of controls available on this computer.

Figure 3. Controls

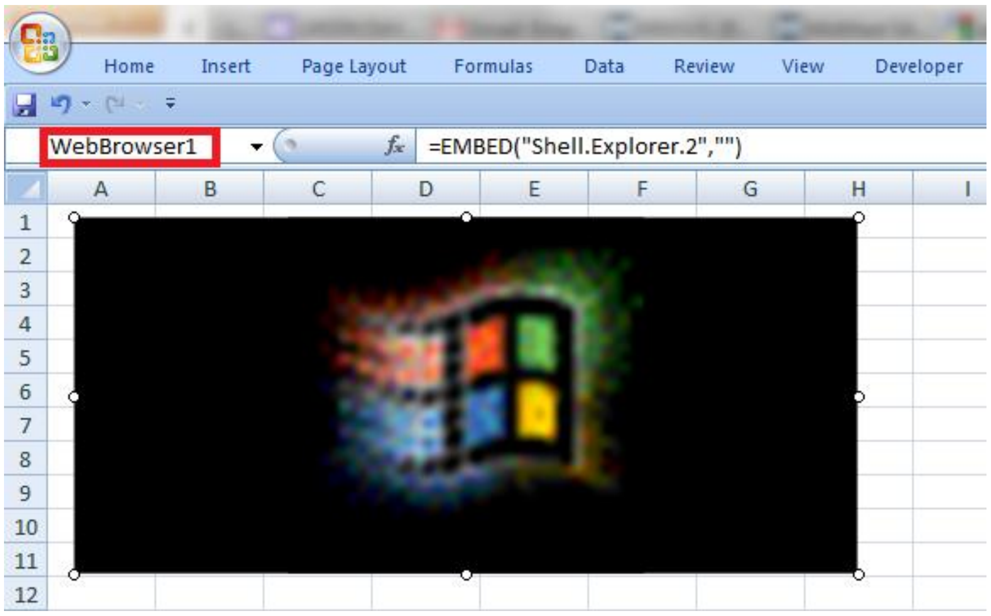
(2) Select “Microsoft Web Browser” -> click OK button (Fig. 4).



Microsoft TabStrip Control, version 6.0
Microsoft Terminal Services Client Control - version 1
Microsoft Toolbar Control, version 6.0
Microsoft TreeView Control, version 6.0
Microsoft Visio Document
Microsoft Web Browser
Microsoft Works Calendar AllDayEvent Control
Microsoft Works Calendar AppControl
Microsoft Works Calendar Banner Control
Microsoft Works Calendar Block Control

Figure 4. More Controls Dialog

(3) Move and size the control to the area you want to display (Fig. 5).
This creates an ActiveX web browser component object and is given the name “WebBrowser1” by default.



=EMBED("Shell.Explorer.2","")

Figure 5. Display area

We can now load any web page using the web browser component in the excel sheet. We will start by writing and loading a simple static html page(gmap.html) which renders a Google map and make use of VBA to load this html in excel using the WebBroser1 object. The HTML sample code is shown in Appendix D. The screenshot for this HTML code is shown in Fig. 6.

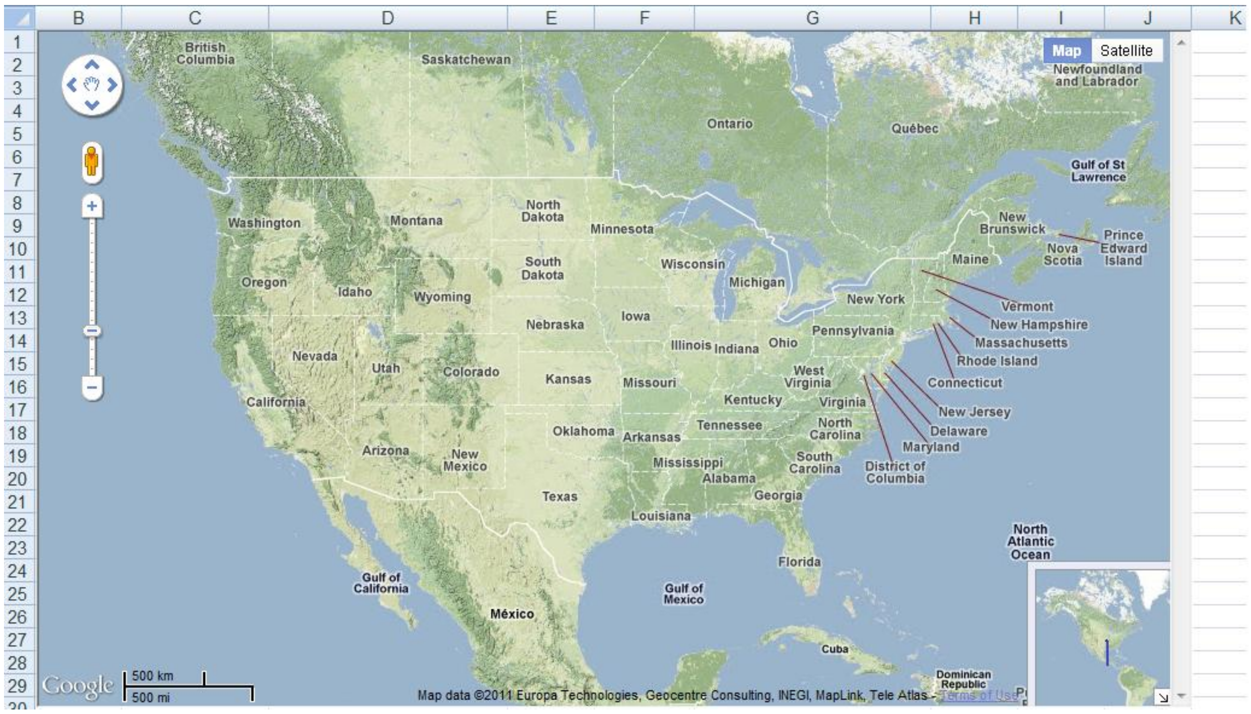

Map
Satellite
Google
500 km
500 mi
Map data ©2011 Europa Technologies, Geocentre Consulting, INEGI, MapLink, Tele Atlas - Terms of Use

Figure 6. Screenshot for HTML Sample

Rendering state-wise target locations with contact information like address, telephone number etc.
The target locations are rendered using markers on the map and the state-wise xml files contain the geospatial information needed to show these location. So we simply parse the XML for the required state and create marker for each target (marker tags in XML) on a map. Internally in the JavaScript, for each marker we also assign a click event which shows a tooltip containing the name, address and mobile number of the target when a user clicks on a marker. The JavaScript code used to add marker on the map is shown in Appendix E with the code for XML parsing in Appendix F and the screenshot is shown in Fig. 7.

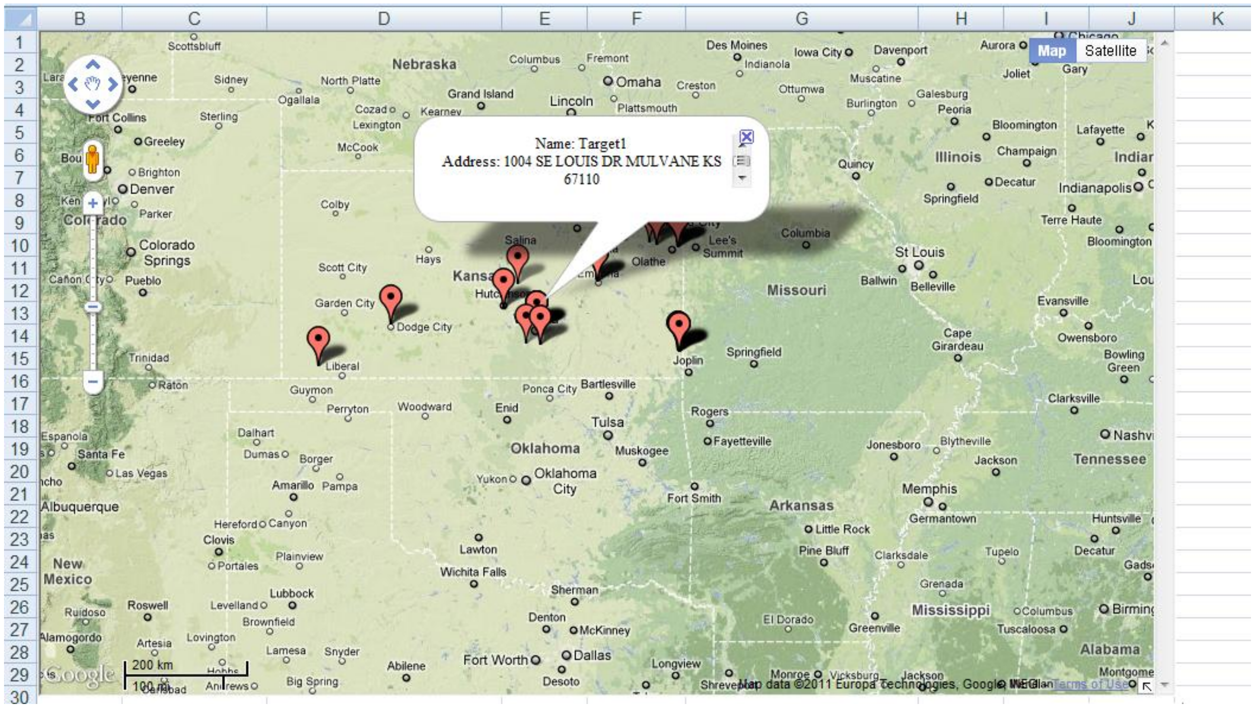


Figure 7. Displaying targets in Map

Writing VBA code for interaction between excel and web browser component:

Since a web browser component object is created in excel ("WebBrowser1") to load a Google map using a custom html file(gmap.html) we can use this object("WebBrowser1") explicitly in the VBA code to call any JavaScript function present in the html. A code example is given below which calls the *removeDirectionsDisplay()* JavaScript function using VBA:

```
WebBrowser1.Document.parentWindow.execScript "parent.removeDirectionsDisplay()"
```

Arguments can also be passed to the JavaScript functions as shown below:

```
WebBrowser1.Document.parentWindow.execScript "parent.calcRoute('DRIVING')”
```

Which passes string "DRIVING" to calcRoute javaScript function.

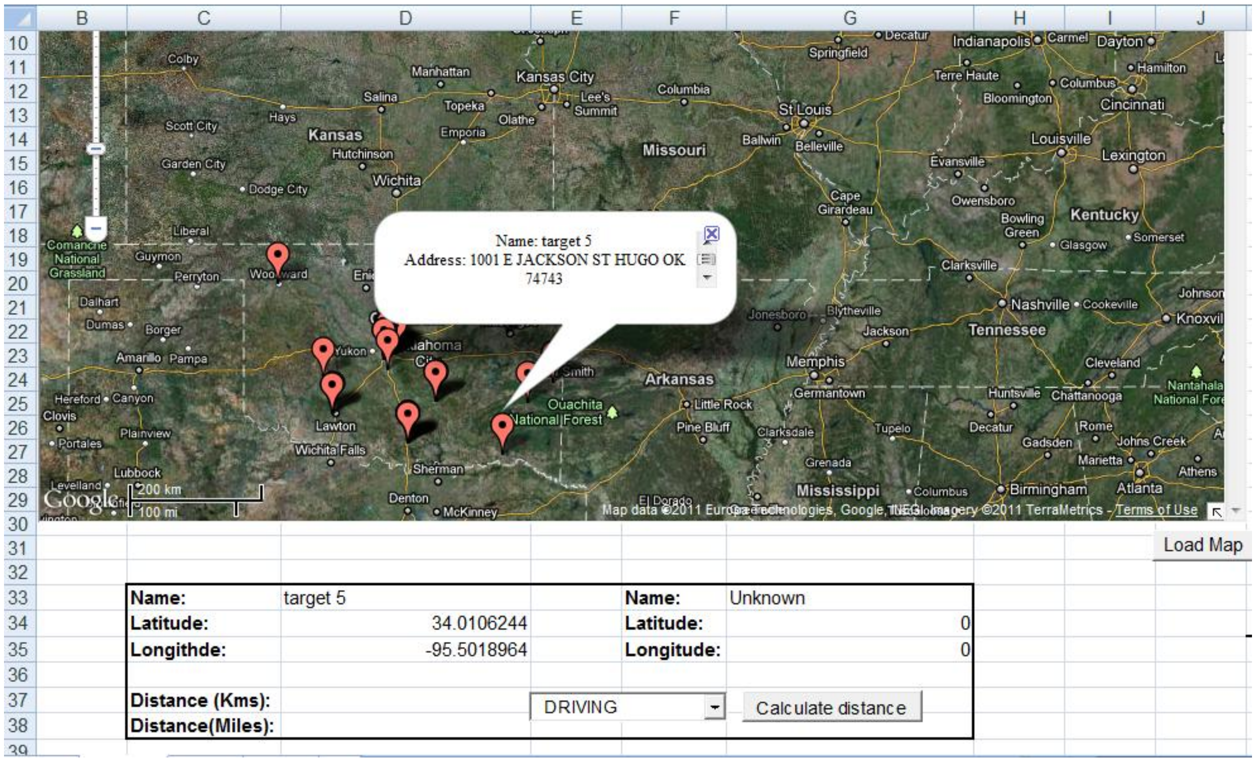
Any data that need to be sent to Excel from the web browser component can be done by creating hidden text boxes in the html page and using VBA to fetch the data from these hidden text boxes. An example is shown below: Suppose the html file shows a text box (Name1) which contains the data which we want to show in a the excel sheet then this can be done using the below VBA code:

```
ActiveWorkbook.Sheets(1).Range("d33") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Name1").Value
```

The VBA code uses the WebBrowser1 object to get the value of textbox Name1 and displays the string in the excel sheet in the d33 cell. JavaScript can be used to trigger VBA code to update the excel sheet by updating text in the web browser status which in turn triggers the *WebBrowser1_StatusTextChange(ByVal Text As String)* event function which can be used to update cells as shown above. JavaScript can change the status of the web browser by using the statement shown below:

Window.status = '<text to show in the status bar>'

Updating target information using JavaScript in excel using VBA.
When a user clicks on a marker shown on a Google map, JavaScript marker click event is triggered which updates the target information in hidden textboxes in the html page and also update the browser status due to which the VBA event WebBrowser1_statusTextChange is triggered. The triggered VBA event function is then used to update the excel sheet. The JavaScript and VBA code which executes when a user clicks on a marker is shown in Appendix G and the screenshot is shown in Fig. 8.



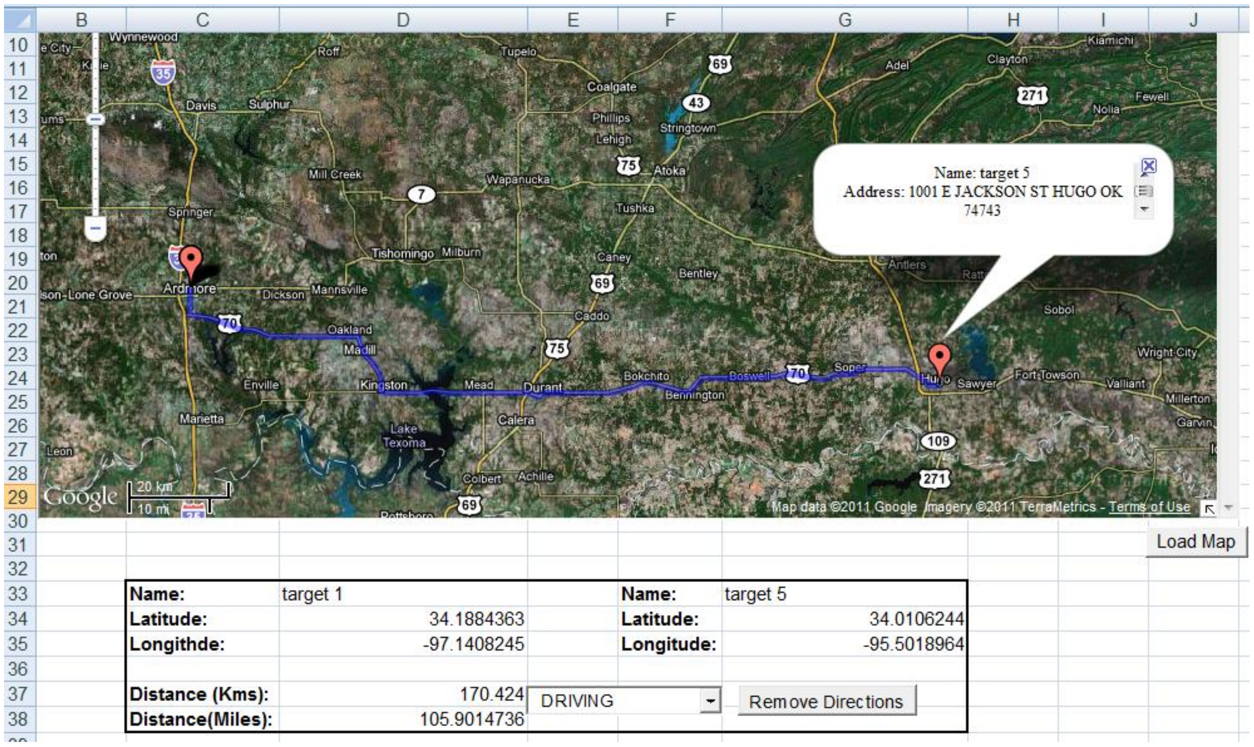
Name: target 5
Address: 1001 E JACKSON ST HUGO OK 74743

Name:	target 5	**Name:**	Unknown
Latitude:	34.0106244	**Latitude:**	0
Longithde:	-95.5018964	**Longitude:**	0
Distance (Kms):			
Distance(Miles):			

DRIVING

Figure 8. Result of a user click on a marker

Calculating the distance between targets and also rendering the path between the two points on the map.
One more service provided by Google is to render the directional path between two locations (markers) and also calculate the distance in meters and miles. After selecting 2 valid markers as shown in the screenshot (Fig. 9) and when the user clicks on the calculate distance button on the excel sheet, it triggers the VBA function CommandButton1_Click() which in turn calls a JavaScript function calcRoute which requests for the calculation of distance between the two markers and also renders the path to be taken to reach the destination. The VBA code for distance calculation is shown in Appendix H and the screenshot for information display is shown in Fig. 9.



Name: target 5
Address: 1001 E JACKSON ST HUGO OK 74743

Name:	target 1	**Name:**	target 5
Latitude:	34.1884363	**Latitude:**	34.0106244
Longithde:	-97.1408245	**Longitude:**	-95.5018964
Distance (Kms):	170.424		
Distance(Miles):	105.9014736		

Figure 9. Information display on user click

Conclusion

In this paper we have seen how Google maps can be integrated with excel with the help of SAS, Java, JavaScript and VBA. This is done though a series of steps listed below:

(1) Getting geocoding information for targets using SAS and java with the help of Google maps web services API and creating state-wise xml file.

(2) Rendering the target locations using markers on a Google map by writing a custom html page

(3) Interaction between html and VBA using the event function (WebBrowser1_StatusTextChange) which gets triggered when the browser status is changed.

(4) Using Google maps services like directional display to render path between targets and calculating distance between the points.

Appendixes

A. SAS Code:

```
LIBNAME temp "C:\Documents and Settings\Adney\Desktop\temp";
DATA targetsGeoData; /* targetsGeoData dataset contains the lat and lng info for each
target*/

       SET temp.targets;


LENGTH lat $40. lng $40.;/* Create variables lat and lng which will hold the
geospatial information */

/* Declare java object j1 */
```

```
DECLARE javaobj j1;
/* Passes target address to java object constructor GeoLoc */
j1 = _NEW_ JavaObj('GeoLoc',street||' '||city||' '||state||' '||zip);

/* Calls GetLatitude java function which returns the latitude of the target address */
j1.callStringMethod('GetLatitude', lat) ;

/* Calls GetLongitude java function which returns longitude of the target address */
j1.callStringMethod('GetLongitude', lng) ;


/* Delete the java object */
j1.Delete();
RUN;
```

B. Java Code

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.MalformedURLException;
import java.net.URL;

/* Using google's JSON library (gson) to parse the response received from Google maps
web service API */
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class GeoLoc {

      private String address;
      private String status;
      private String latitude;
      private String longitude;

      public GeoLoc() {
          this.address = "";
          this.status = "-1";
          this.latitude = "0.0";
          this.longitude = "0.0";
      }

      /* GeoLoc class constructor called by the SAS program */
      public GeoLoc(String streetAddress) {
           this(); /* Function calls the default constructor GeoLoc() to initialize
object properties */
         /* Formatting the address so that it doesn't contain characters like '&', '#'
and ' '
           and replace all spaces with '+' character. */
         streetAddress = streetAddress.replaceAll("&", " ");
         streetAddress = streetAddress.replaceAll("#", " ");
         streetAddress = streetAddress.replaceAll("\\s+", "+");
         this.GetGeoCodes(streetAddress);
      }

      /* Sends an address for geocoding, parses response and loads latitude and
longitude properties. */
      private void GetGeoCodes(String Address) {
           this.address = Address;
```

```
            URL URLsource = null;
            JsonObject jso = null;
            BufferedReader in;
            try {
            /* Creating a valid request to be sent to google maps web services api */
                  String googleBaseUrl =
"http://maps.googleapis.com/maps/api/geocode/json?address=";
                  String sensor = "&sensor=false";
                  System.out.println(googleBaseUrl+this.address+sensor);
                  URLsource = new URL(googleBaseUrl+this.address+sensor);
            /* Request is sent and the response is received in the JSON format */
                  in = new BufferedReader(new
InputStreamReader(URLsource.openStream(), "UTF-8"));
            /* Json Parser object holds the response sent by google which is used for
parsing and getting the required
               information like status of the request('OK', 'REQUEST_DENIED' etc),
latitude and longitude of
               the requested target address. */
                  jso = (JsonObject) new JsonParser().parse(in);
                  in.close();
                  this.status = jso.get("status").getAsString();
            /* If the status is "OK" then load the latitude, longitude and address
properties using Json object*/
                  System.out.println(this.status);
                  if ( this.status.equalsIgnoreCase("OK") ) {
                      latitude =
jso.getAsJsonArray("results").get(0).getAsJsonObject().get("geometry").getAsJsonObject
().get("location").getAsJsonObject().get("lat").getAsString();
                      longitude =
jso.getAsJsonArray("results").get(0).getAsJsonObject().get("geometry").getAsJsonObject
().get("location").getAsJsonObject().get("lng").getAsString();
                      address =
jso.getAsJsonArray("results").get(0).getAsJsonObject().get("formatted_address").getAsS
tring();
                  }
            }
            /* Catch blocks to report any error encountered in the try block */
            catch (MalformedURLException e) {
               e.printStackTrace();
            }
            catch (UnsupportedEncodingException e) {
               e.printStackTrace();
            }
            catch (IOException e) {
               e.printStackTrace();
            }
      }

      /* returns the latitude(as String) of the target address */
      public String GetLatitude() {
            return latitude;
      }
      /* returns the longitude(as String) of the target address */
      public String GetLongitude() {
            return longitude;
      }
      /* returns the parsed address(as String)from JSON fileof the target address */
      public String GetAddress() {
            return address;
      }

      /* returns the status of request sent to google maps api */
      public String GetStatus() {
            return status;
      }
      /* returns the co-ordinates in the format(<latitude>,<longitude>) */
      public String GetCoordinate() {
            return String.valueOf(latitude) + "," + String.valueOf(longitude);
      }
}
```

C. SAS code to create state-wise xml files

```
DATA _NULL_;
   SET temp.targetsGeoData;
   LENGTH fname $100.;
   fname= 'C:\Documents and Settings\Adney\Desktop\temp\'||trim(put(state,$13.)) ||
'.xml';
   FILE datfiles FILEVAR=fname MOD;
   /*FILEVAR=variable option defines a variable whose change in value causes the FILE
statement to close the current output file and open a new one the next time the FILE
statement executes.
     MOD option writes the output lines after any existing lines in the file.*/
   BY state;
   IF first.state THEN DO;
    PUT '<markers>';
   END;
       PUT '<marker lat="' lat +(-1) '" lng= "' lng +(-1) '" name="' target_name '"' /
              'html="&lt;DIV ALIGN=CENTER&gt;&lt;font size=2&gt;' /
              'Name: ' target_name '&lt;br&gt;' /
              'Address: ' street ' ' city ' ' state ' ' zip '&lt;br&gt;' /
              'Telephone: ' target_mob '&lt;br&gt;' /
              '&lt;/font&gt;&lt;/DIV&gt;" />';
   IF last.state THEN DO;
       PUT '</markers>';
   END;
RUN;
```

D. HTML Sample Code:

HTML sample code(gmap.html):

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no"/>
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>Google Maps JavaScript API v3</title>
<link
href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
rel="stylesheet" type="text/css" />
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
  var map;
  function initialize() {
       var myOptions = {
              zoom: 4,
              center: new google.maps.LatLng(39.042939, -95.769657),
              panControl: true,
              zoomControl: true,
              scaleControl: true,
            mapTypeControl: true,
            streetViewControl: true,
            overviewMapControl: true,
            mapTypeId: google.maps.MapTypeId.TERRAIN
        }
        map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
}
</script>
</head>
<body style="margin:0px; padding:0px;" onload="initialize()">
<div id="map_canvas" style="width: 100%; height: 100%;"></div>
</body>
</html>
```

The following VBA code is used to load the gmap.html(google map) in the WebBrowser1 object:
ThisWorkbook.ActiveSheet.WebBrowser1.Navigate "C:\Users\Adney\Downloads\excel_google-maps\gmap.html"

E. Javascript code:

```
function addMarker(location,html,name) {
        var marker = new google.maps.Marker({
       position: location,
       map: map,
        name: name,
        title: name,
        content:html
       });
        google.maps.event.addListener(marker, 'click', function() {
                infowindow.setContent(marker.content);
                infowindow.open(map,marker);
                map.setCenter(marker.position);
      });
}
```

F. XML Parsing:

```
function loadStatePhysicians(stateCode) { // pass the state code to show all the
targets belonging to that state
        var xmlDoc = new ActiveXObject("MSXML.DOMDocument");
        xmlDoc.async = "false";
        xmlDoc.load(stateCode+".xml");
        var markers = xmlDoc.documentElement.getElementsByTagName("marker");
        for (var i = 0; i < markers.length; i++) {
                var lat = parseFloat(markers[i].getAttribute("lat"));
                var lng = parseFloat(markers[i].getAttribute("lng"));
                var name = markers[i].getAttribute("name");
                var point = new google.maps.LatLng(lat,lng);
                var html = markers[i].getAttribute("html");
                var marker = addMarker(point,html,name);  //calls the addMarker function
to load markers(target locations) on the Google map
        }
}
```

G. Code for user clicks on a marker:

```
google.maps.event.addListener(marker, 'click', function() {
                infowindow.setContent(marker.content);
                infowindow.open(map,marker);
```

```
             map.setCenter(marker.position);
             document.getElementById('Name1').value = marker.name;
             document.getElementById('Lat1').value = marker.position.lat();
             document.getElementById('Lng1').value = marker.position.lng();
             window.status='UpdateLoc'
       });
```

VBA code called when the browser status is changed:

```
Private Sub WebBrowser1_StatusTextChange(ByVal Text As String)
If Text = "UpdateLoc" Then
    ActiveWorkbook.Sheets(1).Range("d33") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Name1").Value
        ActiveWorkbook.Sheets(1).Range("d34") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Lat1").Value
        ActiveWorkbook.Sheets(1).Range("d35") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Lng1").Value
        ActiveWorkbook.Sheets(1).Range("g33") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Name2").Value
        ActiveWorkbook.Sheets(1).Range("g34") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Lat2").Value
        ActiveWorkbook.Sheets(1).Range("g35") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Lng2").Value
            WebBrowser1.Document.parentWindow.Status = "Done"
End If
If Text = "UpdateDistance" Then
        ActiveWorkbook.Sheets(1).Range("d37") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Kms").Value
        ActiveWorkbook.Sheets(1).Range("d38") =
ThisWorkbook.ActiveSheet.WebBrowser1.Document.getElementById("Mls").Value
            WebBrowser1.Document.parentWindow.Status = "Done"
End If
End Sub
```

H. Code for distance calculation

VBA code:

```
Dim script As String
script = "parent.calcRoute('" + ThisWorkbook.ActiveSheet.ComboBox1.Value + "')"
        ' VBA calls the calcRoute javascript function using the WebBrowser1 object to
        ' calculate the distance in meters and miles and also to render the path.
        WebBrowser1.Document.parentWindow.execScript script
```

JavaScript code:

```
  function calcRoute(travel_mode) {
       directionsDisplay.setMap(map);
       var start = new google.maps.LatLng(document.getElementById('Lat1').value,
document.getElementById('Lng1').value);
    var end = new
google.maps.LatLng(document.getElementById('Lat2').value,document.getElementById('Lng2
').value);
    var request = {
        origin:start,
        destination:end,
        travelMode: google.maps.DirectionsTravelMode[travel_mode]
              //travelMode: travel_mode
    };
    directionsService.route(request, function(response, status) {
      if (status == google.maps.DirectionsStatus.OK) {
              directionsDisplay.suppressMarkers = true;
        directionsDisplay.setDirections(response);
      }
    });
  }
  function computeTotalDistance(result) {
  var total = 0;
  var myroute = result.routes[0];
  for (i = 0; i < myroute.legs.length; i++) {
    total += myroute.legs[i].distance.value;
  }
  total = total / 1000.
  document.getElementById("Kms").value = total;
  total = total * 0.6214;
  document.getElementById("Mls").value = total;
  window.status='UpdateDistance'
}
```

References

[1] Shane Trahan, Mai Nguyen, Inga Allred, Preethi Jayaram, *Integrating Geocode Data from the Google Map API and SAS/Graph®* (2010), pp. 8-10

[2] Ted Conway, *Get to Your Points: Using SAS® to Build Google Maps* (2009), pp. 1-4

Google maps documentation:
[3] http://code.google.com/apis/maps/documentation/javascript/basics.html
[4] http://code.google.com/apis/maps/documentation/javascript/reference.html
[4] http://code.google.com/apis/maps/documentation/javascript/services.html
[5] http://code.google.com/apis/maps/documentation/geocoding
[6] http://code.google.com/apis/maps/documentation/javascript/basics.html

Acknowledgements

I like to thank Airaha Chelvakkanthan Manickam for presenting the paper in MWSUG-2011 conference.

I would also like to thank Viraj Kumbhakarna for encouraging and guiding me to write this paper and like to thank Neeraj Kumbhakarna for proof reading the paper.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Adney Fernandes
Cognizant Technology Solutions Corporation
Plot No. 26 & 27, MIDC,
Pune Infotech Park,
Hinjewadi, Pune-411027
+91 9049444843
adney.victor@gmail.com