Using Google® API's and Web Service in a CAWI questionnaire

Gerrit de Bolster, Statistics Netherlands, 27 September 2010

1. Introduction

From the survey department of Traffic & Transport in Statistics Netherlands came the question if it's possible to use Google® Maps in their CAWI questionnaire for pin-pointing locations. To determine this, a prototype was developed in which the use of Google® Maps was realised. It appeared that not only the Google® Maps API was needed but also the language API and the Maps Web service. This paper describes how this was done.

I must warn you that this paper is about the technology involved. I tried to explain this technology in a most simple way, but I understand that not everybody is able to grasp my solution. This paper is meant for those who want to do more with BlaiseIS than the basic things and I hope it shows that there are a lot of possibilities to extend BlaiseIS questionnaires with proprietary functionality.

Although they specifically asked to investigate the feasibility of using Google® Maps for searching on locations, the survey department of Traffic & Transport never asked the data collection department to introduce this solution in their questionnaire. One of the reasons they came up with later is that the use of Google® Maps would increment the burden for the respondent filling in the questionnaire.

2. Searching locations

The Internet questionnaire for the survey on road transport from the department of Traffic & Transport contains a lot of questions about locations. The location of the home base of the vehicles is asked as well as the place of departure and destiny of the trips made by these vehicles. Furthermore the respondent must fill in the locations where a load was picked up and where it was delivered. In the current questionnaire the respondent can use for these locations lookups on countries and places. Once a country is selected, the lookup for places (towns, cities) is filtered by this country so only the places within this country will appear in the lookup list. As the survey is on international transport, foreign countries are included. As it was expected that respondents will have trouble to spell the names of the foreign places correctly, a trigram-search was applied for the places. If available the respondent could also provide a zip code of the location asked.

Once the filled in questionnaire is received, the given locations are translated into coordinates (latitude/longitude). For this (automatic) translation a web service from a commercial provider is used. If a location could not be translated using this web-service, it should be done manually. As it appeared this happened more than was expected costing too much human resources. Even in some cases the web-service used was not accurate enough.

For these reasons they asked to investigate if it was possible to produce coordinates in the questionnaire selected by the respondent by using a "maps" interface. Of course, it was obvious to try to use Google® Maps as it is the most well known solution in this field.

3. BlaiseIS questionnaires and JavaScript

Blaise Internet questionnaires are based on the principle of client/server. That means that a respondent is entering data on the client side (with the browser on his own computer) and sending that to the server (in our office). This data is subsequently processed on the server by BlaiseIS components and Active Server

Pages (ASP's) producing an updated screen that is sent back to the client. The way the data is processed is determined mainly by the so-called rules in the Blaise data model. The screen that is sent back to the respondent is written in the universal Internet language HTML, normally enriched with JavaScript.

If the respondent is activating a lookup of any kind, the resulting window (screen) should appear in his browser so he can manipulate it and select the text or code necessary. That means it should appear on the client side and not on the server side. Furthermore, the selected text or code should be written to the input line of the question involved, visible for the respondent in his active screen. In other words: client side.

Within the world of (classic) Internet there is only one global accepted way to make your Internet page have some dynamics: the use of JavaScript. Although today there are other means possible (Java-applets, Flash, Silverlight) JavaScript is still the most used solution. To write data from one window to an input line in another window with JavaScript it is necessary to have 2 things: a pointer to that other window and a name (identification) of the input line. If a window was opened by an action in another window, the pointer to that "parent" window is automatically available (in JavaScript it's called "opener"). But how about the name of the input line? In BlaiseIS every visible field in the questionnaire is given a name at the moment the screen is constructed and sent to the browser of the respondent. It is impossible to get this name before e.g. at design time. The only way to obtain this name is, at the moment the window is opened in the respondents' browser to invoke a JavaScript function to scan the HTML source of the active window. This function was developed already before for another type of search (hierarchical) and stored in a small JavaScript library that is automatically included in every questionnaire using the caption of a (camouflaged) label in the Blaise Menu File (.bmf). The function is invoked by a JavaScript command that is activated the moment an image is loaded at the opening of a window. This command is included in the question text of a separate question in the Blaise sources that has only one visible part: an image in the shape of a "Search" button!

The same function is used to give that "Search" button an index number so it is included in the so-called Tab-order. That is the order in which the parts of a browser window are focussed using the Tab key. Once it is focussed it can also be activated by a key and not only by the mouse.

4. Create your own search facility

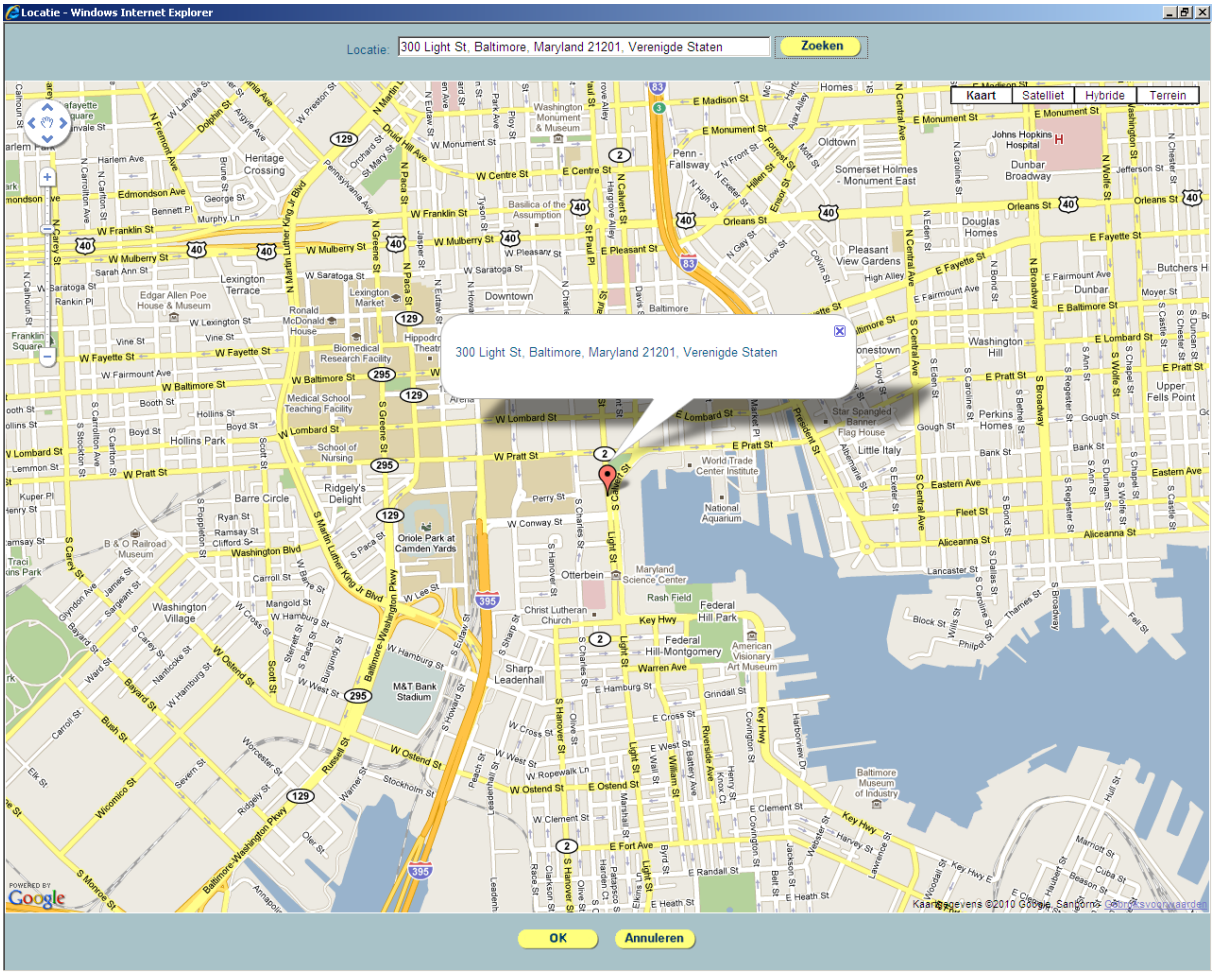
So how do you create your own search facility for your BlaiseIS questionnaire? As was described in the previous section, using JavaScript you can open a new window with the possibility to write data to an input line in the active (visible) page of the questionnaire. This window is opened by clicking on a button image in the questionnaire. As a result it is opened referring to the URL of that window. This URL can e.g. be an URL of an Active Server Page (ASP). In an ASP you can run a script (VBScript by default) invoking Blaise(IS) or other installed components. By the way, BlaiseIS consists partially of ASP's. To create your own search function, you must therefore create your own ASP and include it in your BlaiseIS questionnaire adding it to the file section of the Blaise Internet Specification file (.bis). In this ASP you can manipulate data as much as you want, e.g. opening a Blaise database on the data server and reading records from it or writing records to it. Applying this concept already several special (but generic!) functions have been created including the location search with the Google® Maps API.

As can be understood it is necessary to have the use of JavaScript switched to "on" in the browser if you want to use this special search function. Fortunately almost everybody has JavaScript switched on as without it the Internet is a very dull place. Although Google® still lets you use a simplified form of their Maps facility, it is a rigid solution that is very difficult to use. Just in case, if a respondent is one of the remaining few without JavaScript activated in his browser, he is informed that this search function is not working.

5. Google® Maps API

On the Internet there is a lot of information available regarding the implementation of Google® Maps functionality in a web application. See the section with references at the end of this paper for the home page of Google® Maps in English (also available in other languages). This was the way we learned how to create the location search with the Google® Maps API in JavaScript. The first location search function was created with Google® Maps API version 2 as it was the supported version at that moment. It was obvious that version 2 would not be the supported version much longer, so the solution was converted to version 3, what is the supported version today. There are a lot of examples available on the website making it quite easy to understand the different commands and functions and how to implement them. Basically the use is for free unless you want to include in a commercial environment. However, you can have a contract with Google® allowing you to use it in a different way. This is described in the Terms Of Service (TOS) also available on the Internet. As this location search has not been taken into production until now, Statistics Netherlands have not yet a contract with Google® for this use at the moment. As I understood we do have a contract with Google® for another type of use: that in publications with a geographical nature.

If the documentation is still not enough to answer all your questions, there is also a very active Internet forum on which Google® developers are trying to help you.



The Google® Maps API is a set of JavaScript commands and functions that enables you to manipulate a map in the same way as it is used on the website of Google® itself. You can use navigation buttons, zoom-buttons and switch from map to satellite and so on. Too much to mention. More important is the fact that you can place a marker on the map and move it over the map. To this marker you can attach a small window with information (address) where the marker is standing on the map. Finally, you can even retrieve this address information as it is stored in an object (cluster of data). This object contains a very complicated tree of address parts. It took quite some time to figure out how to retrieve from the object the information that was needed: the street address, the town and the country. This information can than be written to the input line in the BlaiseIS questionnaire by the respondent by pressing an OK-button that was added to the bottom of the screen.

In some cases the marker could be placed on coordinates that did not produce a valid address. Obviously this was the case in the middle of the ocean. But not only there! Google® appeared to be "political sensitive". Certain regions on our planet where there is still a discussion on-going about ownership and such are also producing empty addresses or non-specific addresses. Examples are Kosovo, Western Sahara and Northern Korea. Maybe with the exception of Kosovo, these regions are normally not involved in answers on questions from our questionnaires. No Dutch trucker has been seen there and they are also not very popular holiday resorts.

6. Languages: a nightmare!

Everything appeared to be very easy. There were hardly any problems to get things working using the examples on the website of Google®. This "happiness" lasted until version 1 of the location search was tested! Moving around the maps of the world all kind of strange characters appeared in the address part of the window. Not all of it, but just parts of the address. This happened although the language parameter was specifically set to Dutch (nl). Depending on the country the addresses were shown in Greek, Chinese, Arab or Cyrillic characters. Quite unreadable for our respondents. Although there are not many transporting firms in The Netherlands providing transport services by road to China, Greece and the Balkan countries are within reach. Furthermore, these foreign addresses were stored in UNICODE. This is a character set using 2 characters for one. Blaise, however, only supports the basic ANSI code set, one character at a time. Writing these addresses to the input line in the BlaiseIS questionnaire resulted in unreadable characters.

But, no panic, Google® also supports an API with which you can translate words and sentences. This nice (also JavaScript) API even includes an auto detect of the language of the text to be translated. However, this auto detect does not work when the text to be translated is a mix of different languages. So the language parameter was set to "local" producing the address including the name of the country in the local character set. It appeared that our good friend Murphy was working for Google® too. Suddenly the addresses were shifted all over the world. Cities in Croatia were moved after translation (according to Google®) to Australia, the city of Athens, Greece was translated into Athens, USA and so on. It was not Google® Maps that gave the wrong addresses; it was the translation API that mixed up things.

The problem was finally solved with a mixture of functions, a combined approach. In Google® Maps the language was set back to Dutch producing an address in different languages. In the translation command the language of the text (address) to be translated was set to the specific language based on the country code retrieved from the address object. Not all the countries were included, only those with different character sets. Even after this change some countries with a small number of different characters (Slovenia, Croatia, etc.) kept producing strange translations of addresses. Finally these countries were removed from the translation set and a function converting their "special" characters to characters within the ASCII character set was added. As a last resort a final addition was made. In some rare cases it could happen that there were still some "strange" characters left over (missed by all the other conversions). In

that case the coordinates were retrieved from the address object and written back to the input line of the questionnaire.

It worked! The language nightmare was over.

7. Google® Web service

As was mentioned in section 2 the goal was to retrieve coordinates, not addresses. The address object produced by Google® Maps contains these coordinates (latitude/longitude). But respondents are not really trained in reading coordinates. Therefore, in the questionnaire the location produced by Google® Maps should be a readable address. Besides, in Internet questionnaires the search option is optional. If a respondent is sure of the address he can decide just to type it in without invoking the search function. The conversion to coordinates should therefore take place after the question has been filled in. In other words: on the server in our office.

Besides the interactive Maps API, Google® also provides a web service to convert addresses to coordinates. A link to more information about this web service can be found on the homepage of Google® Maps (see Section 10 References). A web service can simply be invoked by calling its URL with some parameters. It normally responds by sending back an xml-stream. To call an URL from a BlaiseIS questionnaire ca be done using a so-called HTTP-request. For that purpose an Alien procedure was created calling a .NET component. In a Microsoft® environment a class is available for this purpose. This is how it looks:

```
<Runtime.ComVisible(True)> _
    Public Sub GetResp(ByVal pURL As BlAPI3A.Field,
      ByVal pParam As BlAPI3A.Field, ByVal pResp As BlAPI3A.Field,
      ByVal pStatus As BlAPI3A.Field)
        Dim objRequest As New WinHttp.WinHttpRequestClass
        objRequest.Open("GET", pURL.Text + pParam.Text, False)
        objRequest.Send()
        pResp.Text = objRequest.ResponseText
        pStatus.Text = objRequest.StatusText
        objRequest = Nothing
    End Sub
```

By making the question in which the address from the search function is written (or directly typed in by the respondent) so-called “critical”, the rules on the server are invoked when this question looses the focus. In these rules the web-service is called through the Alien procedure and returns an xml stream in a Blaise string field containing the coordinates. This string is subsequently “ripped” with standard Blaise commands to get hold of the coordinates. Using a status field warnings or errors can be given when an address could not be converted. As it is server based, the call to the web service is going outside to the Internet. The fire-walls should therefore be configured to let this call pass through.

8. Look and feel

Using the functions of the Google® Maps API the search function is made as user friendly as possible. A respondent can double click on any place on the map and the marker jumps to that place showing the address in the info window. It is also possible to “grab” the marker with the mouse and drag it to a new place. Furthermore, by clicking on the search button on the top of the search screen, the screen is centred on the marker wherever it is situated. The whole map can be moved by dragging it with the mouse or using the navigation buttons. As Google® Maps is a very used function on the Internet, it looks quite familiar and respondents will probably not have any problem using it.

Finally the search window is given the same standard look and feel of the other BlaiseIS search functions as we designed them at Statistic Netherlands.

9. More functions

The same technology that is used for the location search is applied for other search functions too. At this moment a hierarchical search function is available which is designed for large coding tables like the Combined Nomenclature as used in the European Intrastat survey. Lately a keyword search was added that can e.g. be used for coding occupations. The concept of this popup search function makes it possible to develop new functions in a short time. Finally based on the same concept but applied in a different way a function for a popup question was added to the "family" that can also be used as an alternative for the remark function.

10. References

Google® Maps homepage: http://code.google.com/intl/en/apis/maps/index.html

Google® Maps terms of service: http://code.google.com/intl/en/apis/maps/terms.html

Google® Language homepage: http://code.google.com/intl/en/apis/ajaxlanguage/documentation/

Google® Forums homepage: http://www.google.com/support/forum/