Using Google Maps API along with technology .NET

Michał Konarski, Wojciech Zabierowski

Abstract - **This paper describes how Google Maps API interface can be used together with .NET technology, on example of a web portal for drivers.**

Keywords – **Google Maps API, .NET, JavaScript.**

I. INTRODUCTION

I am sure that everyone who use the Internet, has at least once encountered Google Maps service. You have probably checked it to see a satelite picture of Your home or a place You wanted to visit. Basically it is used on companies's private websites to point locations like “You can find us here”. It can be easily achieved also on Your website with Google Maps API. If You want to learn how to do that and how this interface can be used to develop rich web applications, You should read the following paper. It will cover the topic of using Google Maps API together with technology .NET on example of web portal for drivers.

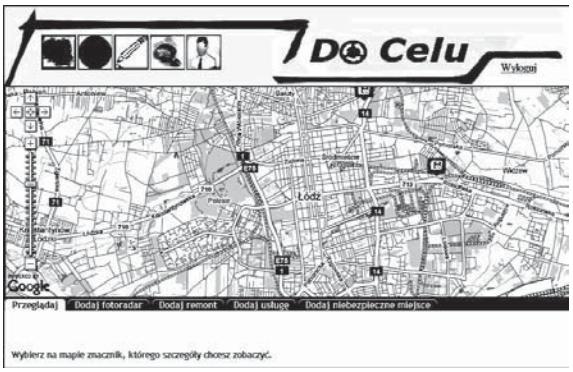
II. GOOGLE MAPS (API)

Google Maps service was presented by Google corporation on February 2005. It was combined with Google Local, which offered relevant neighborhood business listings, maps and directions. From the start, Google Maps service was only available in U.S.A. and Canada. First European country to receive maps and satelite pictures was United Kingdom. Nowadays, it is available in majority of European countries, Poland included. However, high resolutions pictures are only available in big cities, where single trees or buildings can be distinguished. When the service was first presented, it could be only accessed through http://maps.google.com website. As it popularity grown, Goggle company decided to share it with whole world, by creating Google Maps API. It allows developers to integrate Google Maps into their websites with their own data points. It is free to use, but You need to acquire an API key, which is bound to the website and directory entered when creating the key. After having the key generated, we can start building our application.

III. THE CONCEPT

Let’s say that we want to create a portal, where users can place miscellaneous, valuable information for drivers like speed cameras, road repairs, services and dangerous places. It could be then discussed by the community of drivers, so it isn't an object of advertising and would present a honest source of knowledge. This paper isn't about creating a complete portal in .NET technology, but only about using it with Google Maps service. So, I assume that the reader is familiar with ASP.NET and basics of creating web applications and I will focus on using Google Maps API. You can take my suggestions of how your application should look, but You are more than welcome to build a kind of Your own.

The foundation of our portal will be an interface, which will enable the users to browse through the map's content and also place information on it.



Pic. 1

I decided to use the TabContainer control from Ajax Control Toolkit project, because we can place each section of our interface in separate tab and access all of them instantly, without refreshing the page. You can download this control from the project's website (see references, position 3). For example, when the first tab is active, we can browse through the map's data and see detailed information about it. So You should place some labels and textboxes in the tab to handle the information. When the second tab is active, we could add speed cameras. Here, You can place some checkboxes and buttons, to get some information from the user about the speed camera, like if it is active or in which province or town it is placed. A simple button can commit the addition. It’s all up to You. The TabContainer control combined with one object of Google Map will do our interface. An important thing is, that we will load the whole page just once and later do only partial page refreshing by using UpdatePanel controls. This way, we can avoid filling our map with all the data each time a user will generate a postback.

To start, we need to include the Google Maps API script on our asp page.

```
<script src="http://maps.google.com/maps?file=api&amp;amp;v= 2.x&amp;amp;key=..." type="text/javascript"></script>
```

Listing 1

The key attribute value should be the one You have generated before. Next, we need to place the map itself in a div block,

```
<div id="map"></div>
```

Listing 2

and then initialize it in JavaScript with a Gmap2 object and set a starting point:

```
var map = new GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
```

Listing 3

The setCenter function takes a GlatLng object (represents coordinates) and zoom level. When You invoke the above script, You should see Your first map.

IV. FILLING THE MAP

Now that we have our map, we need to fill it with overlays. We will produce a WebService in .NET, responsible for retrieving

them from database and returning them to JavaScript function, which will place them on the map. Database should contain information about the position (latitude and longitude) and outlook of each overlay. Our Webservice will be called OverlayService and it should be declared like this:

```
[System.Web.Script.Services.ScriptService]
public class OverlayService :
 System.Web.Services.WebService
 {
  [WebMethod, ScriptMethod]
  public List<ServiceOverlay> GetDPoints()
  {
    ...
  }
 }
```

Listing 4

It is important to include the ScriptService attribute, so the WebService could be called from JavaScript. Inside it we have an example WebMethod called GetDPoints, which will get all "dangerous places" from database. The implementation of GetDPoints method is up to You, but in the end it should return a List of ServiceOverlay objects.

```
 public class ServiceOverlay
{
    private string _lat;
    private string _lng;
    private string _id;
    private string _outlook;
    private string _type;

    public string lat
    {
       get { return _lat; }
       set { _lat = value; }
    }
    ...
 }
```

Listing 5

where _outlook is a custom overlay icon file name and _type is our portal's overlay type (speed camera, repair, dangerous point,…). The ServiceOverlay object is not made of other complex objects, so it's properties are automatically serialized to JSON format (understood by JavaScript) and we don't have to worry about it.

Next, let's look at JavaScript side of the procedure:

```
 var dpointsArray = [];
 function GetOverlays() {
OverlayService.GetDPoints(InitDPointsArray);
  }
```

Listing 6

First, we may want to declare an array for our overlays. Next, we have a function GetOverlays which should be called when the map is initialized. It invokes the GetDPoints WebMethod and if succeeded, InitDPointsArray function is called.

```
    function InitDPointsArray(result) {
        if (result != null) {
            dpointsArray = result;
        }
        InitOverlays();
    }
```

Listing 7

The result parameter is actually our list of ServiceOverlay objects. After filling the dpointsArray, we can finally call the InitOverlays function, which will fill the map.

V. MARKER MANAGER

Before we continue, I should say a few words about MarkerManager class. It is used to manage visibility of hundreds of markers on a map, based on the map's current viewport and zoom level. Our application is almost certain to have lots of overlays, so it is recommended to use the MarkerManager class, to make it more readable. Another reason for using it, is that it will make our application faster, because all overlays would not be rendered together.

Now, we can get back to InitOverlays function.

```
function InitOverlays()
{
  map=new GMap2(document.getElementById("map"));
  var i = 0;
  map.setCenter(new GLatLng(51.76, 19.52), 12);
  map.addControl(new GLargeMapControl());

  mgr = new MarkerManager(map);
  var batch = [];

   for (i = 0; i < dpointsArray.length; i++)
   {
    var Icon = new GIcon("", "", "");
    Icon.image = dpointsArray[i].outlook;
    Icon.iconSize = new GSize(30, 30);
    Icon.iconAnchor = new Gpoint(0, 30);
    Icon.infoWindowAnchor = new Gpoint(5, 2);
    Icon.transparent = "";
    Icon.shadow = "";

     var point=new GlatLng(
                 dpointsArray[i].lat,
                 dpointsArray[i].lng);
     batch.push(
      CreateMarker(point,
              dpointsArray[i].id,
              Icon,
              dpointsArray[i].type)
            );
   }
     mgr.addMarkers(batch, 10);
     mgr.refresh();
}

function CreateMarker(point, id, Icon ,type) {
     var marker = new GMarker(point,
                    {icon: Icon,});
     marker.value = id;
     return marker;}
```

Listing 8

At the beginning we initialize a GMap2 object, center the map and add a standard map control. Then, we create a MarkerManager object (variable mgr) and a batch array, which holds the overlays for the MarkerManager. Next, we a loop through the dpointsArray, adding markers returned by CreateMarker function to the batch. Our markers have custom icons, represented by GIcon object. Its properties are described on Google Maps API Reference website (see references, position 5). Finally, we fill the mgr variable with batch array, by addMarkers function and refresh it to see the effect.

VI. HANDLING EVENTS

To continue with construction of our user interface, we must handle the events generated by the map and overlays. What we want to do, is to distinguish the events based on active tab of our TabContainer control. Among others, that is because we don't want to add repairs on the map while the speed cameras tab is active and vice versa. When user clicks the map, a map dispatcher should be called:

```
function MapEventDespatcher(point) {
   if (active_tab_index == 0)
   {
                // do nothing
               }
   else if (active_tab_index == 1
   {
    //add speed camera at clicked point
   }
                else if (active_tab_index == 2
    {
    //add repair at clicked point
    }
                ... // and so on
}
```

Listing 9

You can find out a TabContainer control's active tab index through a function that is called on client-side on each tab change. It must be previously defined in a TabContainer declaration.

ASP.NET:

```
<ajaxCT:TabContainer ID="TabCtr1" runat="server"
 OnClientActiveTabChanged
     ="Client_ActiveTabChanged" >
```

JavaScript:

```
function Client_ActiveTabChanged(sender,e) {
active_tab_index=sender.get_activeTabIndex();
}
```

Listing 10

Now, we need to assign the dispatcher to the map's "click" event with GEvent object.

```
GEvent.addListener(map, "click",
  function(marker, point) {
       MapEventDespatch(point);
   });
```

Listing 11

An overlay event dispatcher may look like this:

```
function OverlayEventDespatcher(id) {
   if (active_tab_index == 0)
   {
                // browse clicked overlay
   }
   else if (active_tab_index == 1)
   {
                // remove speed camera
   }
   else if (active_tab_index == 2)
   {
                // remove repair
   }
```

Listing 12

VII. COMMUNICATION

To be able to respond properly to each kind of event on your ASP page, You can use hidden fields of form. For example, if an overlay is clicked, You can save its ID in a hidden textbox and then programmatically press a ASP button from JavaScript:

```
var hidden_txt =
    document.getElementById("hidden_txt");
var hidden_btn =
    document.getElementById("hidden_btn");

if (hidden_txt != null && hidden_btn != null)
{
 hidden_txt.value = ID;
 hidden_btn.click();
}
```

Listing 13

Using hidden field is the most popular way to communicate between JavaScript and .NET. Another possibility is the one mentioned before. You can respond to control's events by defining JavaScript functions that will handle the events. If You would like to call some JavaScript function from code-behind in .NET, You can use the RegisterClientScript method of ScriptManager control, which will invoke the script during page load.

Our portal is using WebService and WebMethods, because it is very fast way of exchanging data and can be done in the background. However, if You want to send some small amount of information from .NET to JavaScript, You can also use ScriptManager control and it's method RegisterArray to register an array on the page and use its values from the script.

VIII. CONCLUSION

When You decide to use Google Maps API in Your application, then You should come up with o good plan of placing the overlays on the map and handling events. It is most of the work You will have to do. As it was shown in this paper, building web applications in .NET technology based on Google Maps service, is not a difficult process. It is basically about the communication between JavaScript and .NET and it can be easily established with usage of a WebService, where data serialization is done automatically.

IX. REFERENCES

http://www.google.com/corporate/history.html ,
http://en.wikipedia.org/wiki/Google_Maps ,
http://www.codeplex.com/Wiki/View.aspx?ProjectName=AjaxControlToolkit ,
http://gmaps-utility-library.googlecode.com/svn/trunk/markermanager/release/docs/reference.html ,
http://code.google.com/intl/pl/apis/maps/documentation/reference.html#GIcon