2012

CycloMedia

GlobeSpotter

[GLOBESPOTTER .NET API / TUTORIAL]

Table of Contents

Support

Please contact:
CycloMedia Technology B.V.
Achterweg 38, 4181 AE Waardenburg, the Netherlands
Postbus 68, 4180 BB Waardenburg, the Netherlands
Tel.: +31 (0)418 556100
Fax: +31 (0)418 556101
e-mail: info@cyclomedia.nl
Dutch Chamber of Commerce (KvK) no.: 16063843

Introduction

This document describes how to use the most basic functions of the .NET version of the GlobeSpotter API in order to add GlobeSpotter functionality to a Geographic Information System (GIS).

About GlobeSpotter

GlobeSpotter is an online environment that enables you to view aerial photos and cycloramas (360° images). Image data and other necessary files are downloaded directly from CycloMedia servers. Because of this, the system can be easily integrated into your daily routine. Besides this, you'll always have access to up-to-date data. GlobeSpotter is suitable to a diverse array of customers, ranging from financial institutions to building contractors to local and national governments.

About the API

The API offers functionality to view and interact with one or multiple cycloramas. Perform measurements and overlay several types of vector data.

Requirements

This version of the API requires:

- A programming language, environment or plug-in system compatible with the .NET framework (version 3.5).
- An internet connection.
- Adobe FlashPlayer 10.

In addition, the implementer will need the following libraries:

- GlobeSpotterAPI.dll
- AbstractAPI.dll
- Interop.ShockwaveFlashObjects.dll
- AxInterop.ShockwaveFlashObjects.dll

Overview

In order to use the API, an understanding of what happens `under the hood' is useful. Some developers may want to skip ahead to the Example section instead.

GlobeSpotter

Please Note that GlobeSpotter is a Flash-application, which means that it can be run from either a local or a remote file. Since an internet-connection will be required in most cases for the services (such as the tile-service, which loads the images from the web tile by tile), it is assumed in this tutorial that the default location (on the CycloMedia servers) will be used.

GlobeSpotter uses some online services, such as (at least one) a panorama service for the actual images and a service that returns the locations of the images (recording locations). Also, in order to function at all, GlobeSpotter needs to know the Spatial Reference System (SRS) of the Cyclorama's. The SRS for cycloramas, the SRS for the address service, the address language code and the API-key ***must*** be set in order for GlobeSpotter to proceed further than starting up Flash.

Please Note that the recording location service and the SRS are unique, so the old one is overwritten each time they are set. However, there can be more than one tile-service active at the same time (although only one is needed for GlobeSpotter to function). Therefore, multiple `set'-actions can be done to the tile-server, without overwriting the old value(s).

Once these parameters are set correctly, the API is ready for use (and it will call a method on the implementation to signify this).

The DLL

The implementer will need to construct an API-object. Once this is created, the implementer can retrieve a GUI-component from this object and add it to the implementing user-interface in the usual manner. Also, the implementer will need to initialize API separately afterwards. This was done in order to be able to give the API-object that was just created to any listener that may need it, before the GlobeSpotter starts firing events.

After the initialization, GlobeSpotter will tell the implementer that the SWF-file has been loaded, by calling *OnComponentReady*. (After initialization, until this moment, a call to any API-method on the object that has just been initialized will give an exception.) In this method, the implementer should set the various application-parameters (with *SetSrsNameViewer*, *SetSrsNameAddress, SetAddressLanguageCode, SetServiceUrl*), such as the services and the SRS-name. Note that the SRS-name and the API-key must be set in order to proceed. Other types of parameter may be set, but have default values. Note that *SetApplicationParameter* calls outside of the scope of *OnComponentReady* are not supported in this version of the API. Only a few other methods (like *GetMajor*- and *GetMinorVersion*) can be called here (any call to a function that can't be called yet will result in an exception). One of the methods that may also be called is *SetUserNamePassword*, which sets user credentials for the session. If this is not done (or not done correctly) GlobeSpotter will ask the user for a password for each significant action.

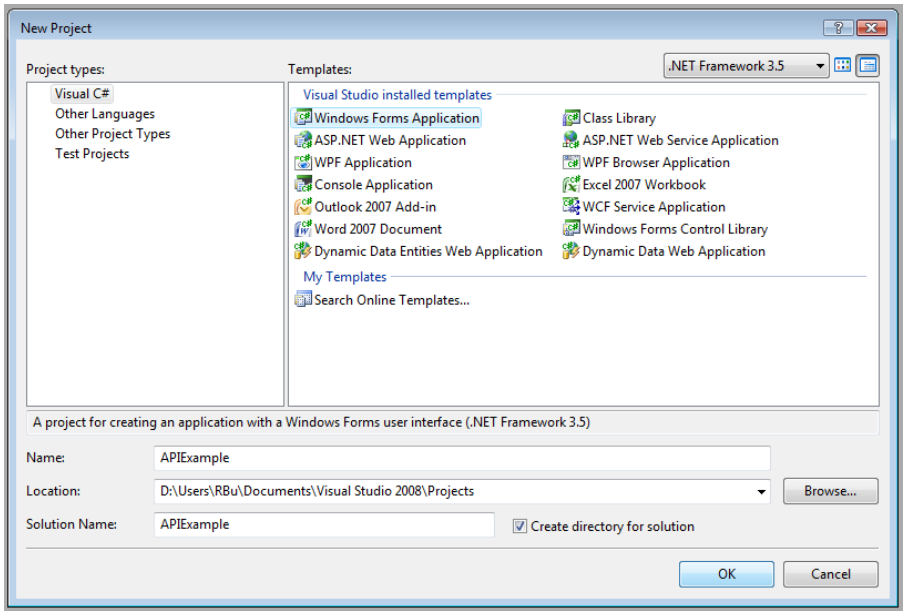
When all required application-parameters are set and valid, GlobeSpotter will tell the application that the API is ready to be called. From this call onward, all functions may be called.

Example

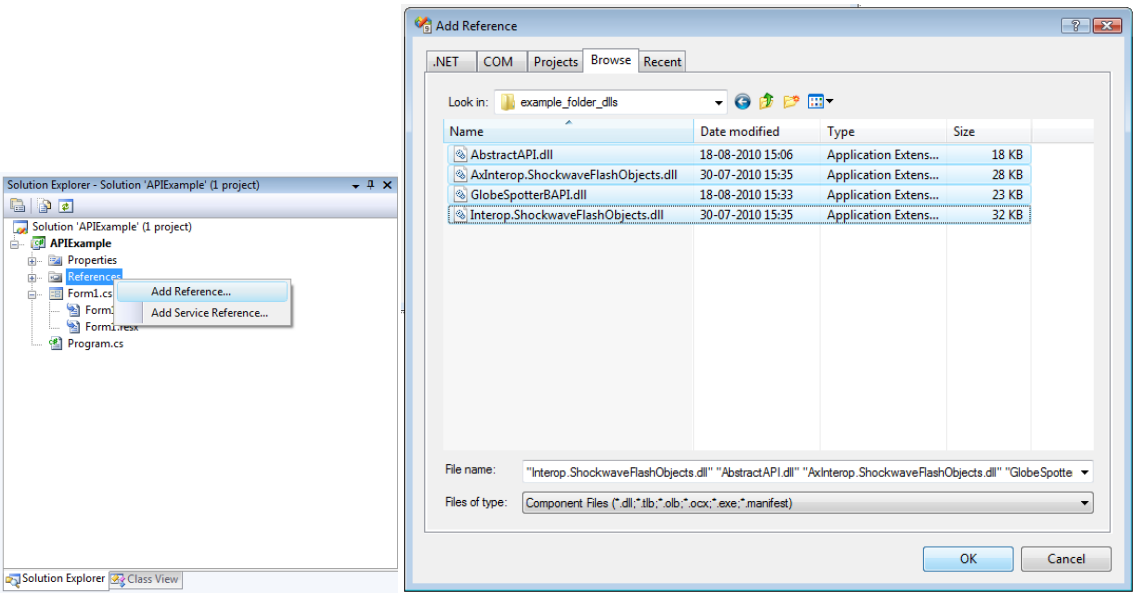
This tutorial guides the implementer through the creation of a simple implementation of the GlobeSpotter API. It assumes the use of Visual Studio 2008, though there should be a way to create something similar in any environment supporting .NET 3.5 or later DLLs. (This section assumes right-handed settings are used for the mouse.)

Project Set-Up

Start Visual Studio, and create a `Windows Forms App' (select: `new project', you will be asked what kind of project is needed). Some, if not most, other project-forms, like the `Windows Presentation Foundation' can also work, but this tutorial assumes Windows Forms is chosen.



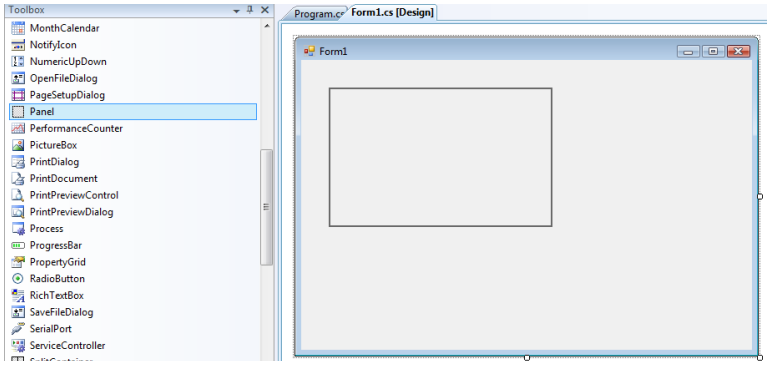
Right-click on `References' in the solution explorer, and add the requisite DLLs.



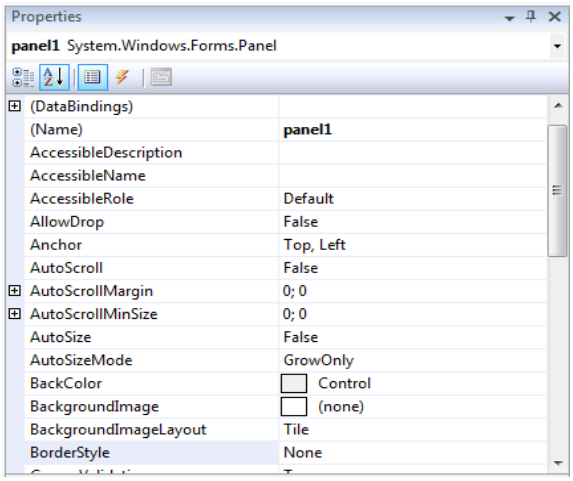
Please note that in Visual Studio 2008, the DLLs will be placed in the output directory once the references are selected. This may not be the case in other versions. In Visual Studio 2010, for example, this is not the default action, so properties need to be set.

GUI Design

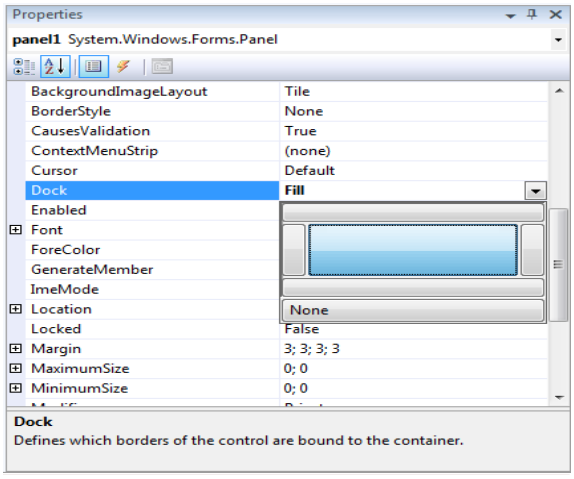
In the Form1 design, create a Panel.



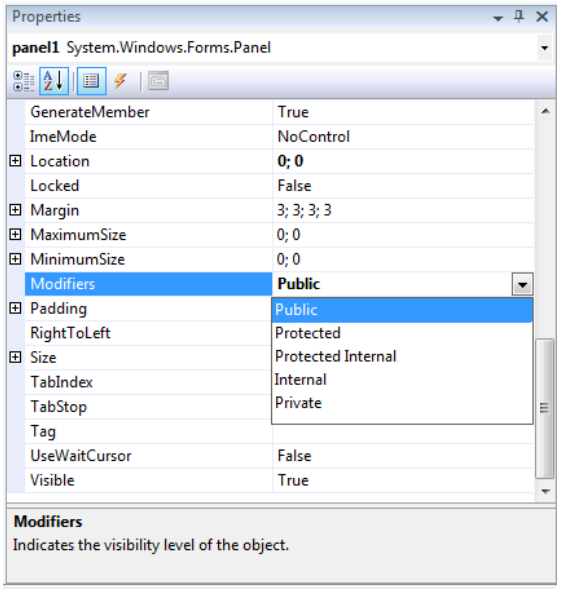
Note the name of the panel `panel1'.



Change `Dock' in the properties of the panel to `fill'.



Change the modifier of the panel to `public' (or `internal').



Write Code

Add `'using GlobeSpotterAPI;'` to the used namespaces above the Program-class.

Remove the 'static' modifier in the Program-class. (Since Program is going to implement a listener of sorts, and this listener needs to be registered as an object, an object of the class Program will be needed.)

Extend the main Program-class with `` `: AbstractAPIClient' `` (or `` `: IAPIClient' ``, if an interface is preferred instead of an abstract class). These contain the methods GlobeSpotter will call on the implementation.

Create the Program-constructor, and move code from the Main()-method to the Program-constructor, so a Program-object can be created, which is also a listener, since it implements a API client.

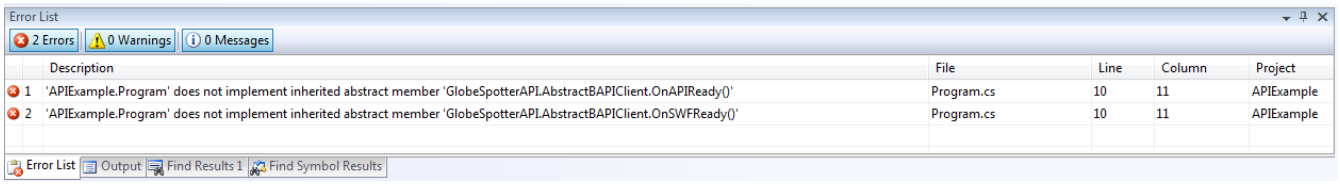
```
//Old situation:
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1());
}

//New situation:
static void Main()
{
    new Program();
}

public Program()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1());
}
```

Add a member-object to the Program-class `API _api;'

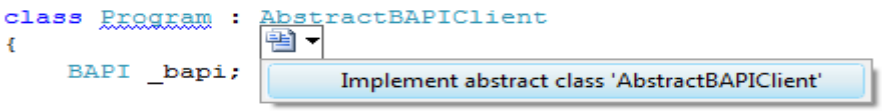
Change the Program-constructor code.

```
public Program()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    // Run the GlobeSpotter component.
    _api = new API(InitType.REMOTE);

    Form1 form = new Form1();
    // Add the component to the Form created by Visual Studio.
    form.panel1.Controls.Add(_api.gui);
    // Tell GlobeSpotter to initialize, as the program is ready to receive events.
    _api.Initialize(this);
    Application.Run(form);
}
```

Try to compile. Note that Program, since it implements AbstractAPIClient, is missing two methods (more if IAPIClient).



Click `Implement abstract class 'AbstractAPIClient", and add the following code, or implement the functions by hand.



First, change OnComponentReady. OnComponentReady is called when GlobeSpotter is ready to receive application-parameters, but GlobeSpotter still needs some of those parameters to become fully initialized.

```
\\Old situation:
public override void OnComponentReady()
{
    throw new NotImplementedException();
}

\\New situation:
public override void OnComponentReady()
{
    // The GetVersionXXX methods are available here too.
    MessageBox.Show("GlobeSpotterAPI version : " + _api.GetMajorVersion() + " ("
        + _api.GetMinorVersion() + ")");

    // Set the API Key, without which it is impossible to proceed.
    _api.SetAPIKey("key_value");

    // EPSG 28992 is the Dutch 'Rijksdriehoekssysteem' SRS.
    _api.SetSrsNameViewer("EPSG:28992");
    _api.SetSrsNameAddress("EPSG:28992");
```

```
    _api.SetAdressLanguageCode("nl");

}
```

Note that ‘SetSrsNameViewer’, ‘SetSrsNameAddress’, ‘SetAddressLanguageCode’, `GetApplicationName’, `GetMajorVersion', `GetMinorVersion', `GetAPIReadyState’ and `SetUserNamePassword’ are the only callable methods of the API-class in this method. Other methods will generate an exception, or do nothing. Of these, ‘SetSrsNameViewer’, SetSrsNameAddress’ and ‘SetUserNamePassword’ are only supported within the scope of ‘OnComponentReady’ in the current version of the API. The last of these (`SetUserNamePassword’) can be used to set user credentials, for a single-point authorization scheme.

Also note that ‘key_value’ is just an example, and will in all probability not work in actual code

Then, change OnAPIReady. This method is called when all necessary parameters are set correctly (in OnComponentReady, or a method called by OnComponentReady).

```
public override void OnAPIReady()
{
    // This will enable the user to navigate via recording locations.
    _api. SetRecordingLocationsVisible(true);

    // Open an image by way of coordinates in the specified SRS
    _api.OpenNearestImage("136833 455816");
}
```

At this point, the code will compile and run. Please try. You will be prompted for your GlobeSpotter password. Move the image by left-clicking, holding and moving. You will probably notice some recording-locations, when rotating. These are clickable, whereupon the user will navigate to the corresponding recording-location.

Next, some information is extracted from GlobeSpotter whenever the user clicks the image. In the Program-class, override the following method, like this:

```
public override void OnViewerClicked(uint viewerID, double[] mouseCoords)
{
    // Retrieve the recording location of the cyclorma.
    RecordingLocation recloc = _api.GetRecordingLocation(viewerID);
    MessageBox.Show("Recording Location: id: " + recloc.Id + " date taken: "
        + recloc.RecordedAt + " x-coord: " + recloc.X + " y-coord: " + recloc.Y);
}
```

This will show the user information on the recording location every time the cyclorama is clicked on. This concludes the tutorial. There are many more functions in even the API, but these are beyond the scope of this tutorial.

The Entire Example Code

This section presents the code of the Program-class. (Note that Form1 is needed, as generated by Visual Studio, with an added Panel.) Also note that the namespace used is `APIExample’, please check if this is the case in any Visual Studio-generated code as well, or adapt either

code to reflect this fact. Thirdly, note the ‘key_value’ that is given as an API-key is just an example, and will not work in production code. **Finally, this code assumes the references and the GUI are set up correctly in Visual Studio.**

```
using System;
using System.Windows.Forms;

using GlobeSpotterAPI; // Enables use of the GlobeSpotter API, given correct references.

namespace APIExample
{
    class Program : AbstractAPIClient // ‘IAPIClient’ also works, but needs more functions.
    {
        API _api; // The object used for access to GlobeSpotter.

        [STAThread]
        static void Main()
        {
            new Program();
        }

        public Program()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            _api = new API(InitType.REMOTE); // Runs the GlobeSpotter component.

            Form1 form = new Form1();
            form.panel1.Controls.Add(_api.gui);

            _api.Initialize(this); // Integration is ready to receive events.
            Application.Run(form);
        }

        public override void OnComponentReady()
        {
            // The GetVersionXXX methods are available too.
            MessageBox.Show
                ( "GlobeSpotterAPI version : "
                + _api.GetMajorVersion() + " (" + _api.GetMinorVersion() + ")"
                );

            // Set the API Key, without which it is impossible to proceed.
            _api.SetAPIKey("key_value");

            // EPSG 28992 is the Dutch 'Rijksdriehoekssysteem' SRS.
            _api.SetSrsNameViewer("EPSG:28992");
            _api.SetSrsNameAddress("EPSG:28992");
            _api.SetAdressLanguageCode("nl");
        }

        public override void OnAPIReady()
        {
            // This will enable the user to navigate with the help of recording locations.
            _api. SetRecordingLocationsVisible(true);

            // Open an image by way of coordinates in the specified SRS
            _api.OpenNearestImage("136833 455816");
        }

        public override void OnViewerClicked(uint viewerID, double[] mouseCoords)
        {
            RecordingLocation recloc = _api.GetRecordingLocation(viewerID);

            MessageBox.Show
```

```
                ( "Recording Location: id: " + recloc.Id + " date taken: " + recloc.RecordedAt
                + " x-coord: " + recloc.X + " y-coord: " + recloc.Y
                );
        }
    }
}
```

Contents of the Core API Package

API_NET_Tutorial.pdf	This document.
API_doc.cmh	Compiled HTML help file.
GlobeSpotterAPI.dll	DLL with all functions of the API.
GlobeSpotterAPI.xml	Documentation in XML format (for IntelliSense).
AbstractAPI.dll	DLL with supporting functions for the API.
AbstractAPI.xml	Documentation in XML format (for IntelliSense).
Interop.ShockwaveFlashObjects.dll	DLL needed for Flash-support in .NET.
AxInterop.ShockwaveFlashObjects.dll	DLL needed for Flash-support .NET.

Please note that the .xml-files are not strictly necessary; however, they do provide IntelliSense-help for Visual Studio when implementing the API. Nor are the SWF files, as long as they are available online at the CycloMedia servers.