Archived:PyS60 Google Maps API

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

This simple module is based on an example written in J2ME that can be found here: Google Static Maps API in Java ME.

This simple library have the following features:

- geocode addresses to their geographic coordinates
- retrieve static images with given custom size, format and zoom

Get your own Google Maps API Key

NOTE: Usage of this code with the free Google Maps API Key breaks Google's Terms and Conditions (section 10.8). You should purchase an Enterprise License if you wish to use the Google Maps API as shown in this example.

Like the J2ME example you should get your own Google Maps API Key. Instructions here: How to use Google Static Maps data in mobile applications

Source code: GoogleMaps python class

This code aggregates all methods in the J2ME example, including the method **adjust()**, so you can assume the same for the J2ME version, copied above:

If you need to scroll your map, you'll need to calculate a new center for your static image. The following **adjust()** method will return the new map center latitude and longitude, accepting the following arguments:

- the current center **latitude** and **longitude** coordinates
- the **deltaX** and **deltaY**, in pixels, of new map center
- the map **zoom** level

Original code, in JavaScript, is available here: http://home.provide.net/~bratliff/adjust.js

```
#!/usr/bin/env python
#
#       gmaps.py
#
#       Copyright 2008 Ramiro Batista da Luz <ramiroluz@gmail.com>
#
#       This program is free software; you can redistribute it and/or modify
#       it under the terms of the GNU General Public License as published by
#       the Free Software Foundation; either version 2 of the License, or
#       (at your option) any later version.
#
#       This program is distributed in the hope that it will be useful,
#       but WITHOUT ANY WARRANTY; without even the implied warranty of
#       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#       GNU General Public License for more details.
#
#       You should have received a copy of the GNU General Public License
#       along with this program; if not, write to the Free Software
#       Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#       MA 02110-1301, USA.
'''
    gmaps.py: GoogleMaps for pyS60.

    This simple module is based on an example written in J2ME that can be found
    here: [[Google Static Maps API in Java ME]].

    The features are:
      - geocodeaddress
      - static images with custom size, format and zoom.

    The urllib for python have some methods, so we do not need to implement,
    urlEncode and loadHttpFile, the python string type already have a method
    split, so we do not have to implement it too.

    # First import the class GoogleMaps.
    from gmaps import GoogleMaps
    # You must have an API_KEY, and create a object.
    gmap = GoogleMaps('API_KEY')
    # To find the geocode address use the method geocodeAddress.
    gAddress = gmap.geocodeAddress("Leicester Square, London")
    # To retrive an image use the method retrieveStaticImage.
    gImage = gmap.retrieveStaticImage(320, 240, 51.510605, -0.130728, 8, "png32")
    # To get a upper view off the image, delta_x = 100 and delta_y = 0, same zoom=8.
    tMap = gmap.adjust(gAddress[0],gAddress[1],100,0,8)
    # Retrieve the image again.
    gImage = gmap.retrieveStaticImage(320, 240, t[1], t[0], 8, "png32")
'''

import math, urllib

import graphics

class GoogleMaps:
    def __init__(self, api_key):
```


(C) Copyright Nokia 2013. All rights reserved.

```
        self.api_key = api_key
        self.offset = 268435456
        self.radius = self.offset / math.pi
        self.tmp_file = u'e:\\teste.png'

    def geocodeAddress(self, address):
        res = urllib.urlopen(self.getGeocodeUrl(address)).read()
        data = res.split(',')
        if data[0] != "200":
            errorCode = int(data[0])
            raise Exception("Google Maps Exception: %s" %
                            self.getGeocodeError(errorCode))
        else:
            # In csv format the information retrieved is as follows:
            # 1. HTTP status code, (index 0 in python lists).
            # 2. accuracy (See accuracy constants) (index 1 in python lists).
            # 3. latitude (index 2 in python lists).
            # 4. longitude (index 3 in python lists).
            return [float(data[2]),float(data[3])]

    def retrieveStaticImage(self, width, height, lng, lat, zoom, format):
        urllib.urlretrieve(self.getMapUrl(width, height,
                                          lng, lat, zoom,
                                          format), self.tmp_file)
        return graphics.Image.open(self.tmp_file)

    def getGeocodeError(self, errorCode):
        dictError = {400: "Bad request",
                     500: "Server error",
                     601: "Missing query",
                     602: "Unknown address",
                     603: "Unavailable address",
                     604: "Unknown directions",
                     610: "Bad API key",
                     620: "Too many queries"}

        return dictError.get(errorCode, "Generic error")

    def getGeocodeUrl(self, url):
        urlbase = "http://maps.google.com/maps/geo"
        return "%(urlbase)s?%(params)s" % {'params': urllib.urlencode({'q':url,
                                           'output':'csv',
                                           'key':self.api_key
                                          }),
             'urlbase': urlbase}

    def getMapUrl(self, width, height, lng, lat, zoom, format):
        urlbase = "http://maps.google.com/staticmap"
        params = ["center=%(lat)s,%(lng)s" % {"lat":lat,"lng":lng}]
        params.append("format=%(format)s" % {"format":format})
        params.append("zoom=%(zoom)s" % {"zoom":zoom})
        params.append("size=%(width)sx%(height)s" % {"width":width,"height":height})
        params.append("key=%(api_key)s" % {"api_key":self.api_key})
        return  "%(urlbase)s?%(params)s" % {"urlbase":urlbase,"params":"&".join(params)}

    def adjust(self, lat, lng, deltaX, deltaY, z):
        return (self.XToL( self.LToX(lat) + (deltaX<<(21-z))),
               self.YToL(self.LToY(lng) + (deltaY<<(21-z))))

    def LToX(self, x):
        return round(self.offset + self.radius * math.radians(x))

    def LToY(self, y):
        return round( self.offset - self.radius *
            (math.log((
            (1 + math.sin(math.radians(y)))
            /
            (1 - math.sin(math.radians(y)))
            )
        )) / 2)

    def XToL(self, x):
        return math.degrees((round(x) - self.offset) / self.radius)

    def YToL(self, y):
        return math.degrees(math.pi / 2 - 2 * (
                math.atan(
                    math.exp(((round(y)-self.offset)/self.radius))
                )
            ))
```

Source code: sample usage

Use the following code to test the API, YOU HAVE TO CHANGE THE API_KEY to your key.

```
# First import the class GoogleMaps.
from gmaps import GoogleMaps
# You must have an API_KEY, and create a object using your API_KEY.
gmap = GoogleMaps('API_KEY')
# To find the geocode address use the method geocodeAddress.
gAddress = gmap.geocodeAddress("Leicester Square, London")
# To retrive an image use the method retrieveStaticImage.
gImage = gmap.retrieveStaticImage(320, 240, 51.510605, -0.130728, 8, "png32")
# To get a upper view off the image, delta_x = 100 and delta_y = 0, same zoom=8.
tMap = gmap.adjust(gAddress[0],gAddress[1],100,0,8)
# Retrieve the image again.
gImage = gmap.retrieveStaticImage(320, 240, tMap[1], tMap[0], 8, "png32")
gImage.save(u"e:\\gmaps_51.51_-1.13.png")
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.