



Here is a simple library to query Google Maps with the following features:

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

To see a live sample of this API, you can check here: [Java ME Google Maps API sample MIDlet](#)

Contents

- [1 Get your own Google Maps API Key](#)
- [2 Use a Proxy server to access Google Map services](#)
- [3 Source code: GoogleMaps class](#)
- [4 Utility method for map scrolling](#)
- [5 Source code: sample usage](#)

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.

To use the following code, you should get your own Google Maps API Key. If you have not an API key, you can follow the instructions here: [How to use Google Maps data in mobile applications](#)

Use a Proxy server to access Google Map services

Note: this subject (proxy usage) is probably non necessary, still investigating it..

When you sign up to obtain a Google Maps API key, you enter the address that will be able to access Maps services with that key. For this reason, you should setup a Proxy server on that address that will receive HTTP requests from your mobile clients, forwarding them to Google Maps services, giving back Google responses.

In the code below, you should forward the following requests:

- <http://www.yourserver.com/geo> requests to <http://maps.google.com/maps/geo>
- <http://www.yourserver.com/staticmap> requests to <http://maps.google.com/staticmap>

Source code: GoogleMaps class

```
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Vector;
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
```

Google_Maps_API_in_Java_ME

```
import javax.microedition.lcdui.Image;

public class GoogleMaps {
    private static final String URL_UNRESERVED =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
        "abcdefghijklmnopqrstuvwxyz" +
        "0123456789-_.~";
    private static final char[] HEX = "0123456789ABCDEF".toCharArray();

    // these 2 properties will be used with map scrolling methods. You can remove them if not needed
    public static final int offset = 268435456;
    public static final double radius = offset / Math.PI;

    private String apiKey = null;

    public GoogleMaps(String key) {
        apiKey = key;
    }

    public double[] geocodeAddress(String address) throws Exception {
        byte[] res = loadHttpFile(getGeocodeUrl(address));
        String[] data = split(new String(res, 0, res.length), ',');

        if (data[0].compareTo("200") != 0) {
            int errorCode = Integer.parseInt(data[0]);
            throw new Exception("Google Maps Exception: " + getGeocodeError(errorCode));
        }

        return new double[] {
            Double.parseDouble(data[2]), Double.parseDouble(data[3])
        };
    }

    public Image retrieveStaticImage(int width, int height, double lat, double lng, int zoom,
        String format) throws IOException {
        byte[] imageData = loadHttpFile(getMapUrl(width, height, lng, lat, zoom, format));

        return Image.createImage(imageData, 0, imageData.length);
    }

    private static String getGeocodeError(int errorCode) {
        switch (errorCode) {
            case 400:
                return "Bad request";
            case 500:
                return "Server error";
            case 601:
                return "Missing query";
            case 602:
                return "Unknown address";
            case 603:
                return "Unavailable address";
            case 604:
                return "Unknown directions";
            case 610:
                return "Bad API key";
            case 620:
                return "Too many queries";
            default:
                return "Generic error";
        }
    }
}
```

Google_Maps_API_in_Java_ME

```
private String getGeocodeUrl(String address) {
    return "http://maps.google.com/maps/geo?q=" + urlEncode(address) + "&output=csv&key="
        + apiKey;
}

private String getMapUrl(int width, int height, double lng, double lat, int zoom, String format) {
    return "http://maps.google.com/staticmap?center=" + lat + "," + lng + "&format="
        + format + "&zoom=" + zoom + "&size=" + width + "x" + height + "&key=" + apiKey;
}

private static String urlEncode(String str) {
    StringBuffer buf = new StringBuffer();
    byte[] bytes = null;
    try {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        dos.writeUTF(str);
        bytes = bos.toByteArray();
    } catch (IOException e) {
        // ignore
    }
    for (int i = 2; i < bytes.length; i++) {
        byte b = bytes[i];
        if (URL_UNRESERVED.indexOf(b) >= 0) {
            buf.append((char) b);
        } else {
            buf.append('%').append(HEX[(b >> 4) & 0x0f]).append(HEX[b & 0x0f]);
        }
    }
    return buf.toString();
}

private static byte[] loadHttpFile(String url) throws IOException {
    byte[] byteBuffer;

    HttpConnection hc = (HttpConnection) Connector.open(url);
    try {
        hc.setRequestMethod(HttpConnection.GET);
        InputStream is = hc.openInputStream();
        try {
            int len = (int) hc.getLength();
            if (len > 0) {
                byteBuffer = new byte[len];
                int done = 0;
                while (done < len) {
                    done += is.read(byteBuffer, done, len - done);
                }
            } else {
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                byte[] buffer = new byte[512];
                int count;
                while ( (count = is.read(buffer)) >= 0 ) {
                    bos.write(buffer, 0, count);
                }
                byteBuffer = bos.toByteArray();
            }
        } finally {
            is.close();
        }
    } finally {
        hc.close();
    }
}
```

```

    }

    return byteBuffer;
}

private static String[] split(String s, int chr) {
    Vector res = new Vector();

    int curr;
    int prev = 0;

    while ( (curr = s.indexOf(chr, prev)) >= 0 ) {
        res.addElement(s.substring(prev, curr));
        prev = curr + 1;
    }
    res.addElement(s.substring(prev));

    String[] splitted = new String[res.size()];
    res.copyInto(splitted);

    return splitted;
}
}

```

Utility method for map scrolling

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://www.polyarc.us/adjust.js>

Note: to use the following methods, you must include in your project the **MicroFloat** library, available here: [MicroFloat website](#)

```

public double[] adjust(double lat, double lng, int deltaX, int deltaY, int z)
{
    return new double[]{
        (LToX(lng) + XTDeltaX<<(21-z))),
        (LToY(lat) + YTDeltaY<<(21-z))
    };
}

double LToX(double x)
{
    return round(offset + radius * x * Math.PI / 180);
}

double LToY(double y)
{
    return round(
        - radius * offset
        Double.longBitsToDouble(MicroDouble.log(
        Double.doubleToLongBits(

```

Google_Maps_API_in_Java_ME

```
(1 + Math.sin(y * Math.PI / 180))
/
(1 - Math.sin(y * Math.PI / 180))
)
)) / 2);
}

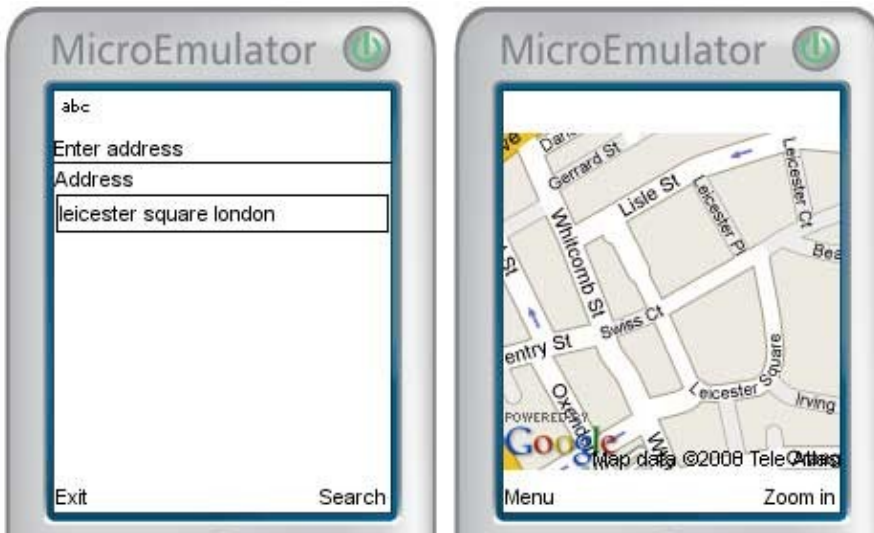
double XToL(double x)
{
return ((round(x) - offset) / radius) * 180 / Math.PI;
}

double YToL(double y)
{
return (Math.PI / 2 - 2 * Double.longBitsToDouble(
        atan(
            MicroDouble.
                exp(Double.doubleToLongBits(round(y) - offset) / radius))
        )
) * 180 / Math.PI;
}

double round(double num)
{
double floor = Math.floor(num);

if(num - floor >= 0.5)
return Math.ceil(num);
else
return floor;
}
}
```

Source code: sample usage



To use this class, you firstly instantiate it with your API key:

```
GoogleMaps gMap = new GoogleMaps("API_KEY");
```

To geocode an address, you can use the geocodeAddress() method:

```
double[] lanLng = gMap.geocodeAddress("Leicester Square, London");
```

Google_Maps_API_in_Java_ME

To retrieve a map image:

```
Image map = gMap.retrieveStaticImage(320, 240, 51.510605, -0.130728, 8, "png32");
```