



Hi all!! In this post i will show you an interesting application with Location API. The goal is to get the GPS position (with Location API - JSR 179) and plot the position with Google Maps.

The Location API (JSR 179)

The Location API (JSR 179 - an optional package) javax.microedition.location aims at helping the development of location-based applications and services for resource-limited devices, like mobile phones. The application must be set for CLDC 2.0 (CLDC 1.0) does not support floating-point number, which the API uses to represent coordinates and other measurements).

The Google Map API

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.

The Google Map API allows you embed Google Maps in your own web pages with JavaScript. Therefore, we will just use a web services REST to get the map image and plot in the device's screen. We just must send a request for <http://maps.google.com/staticmap?params> URL and set some parameters, such as, latitude and longitude coordinates, image's size, zoom, map type (mobile, etc) and, if you wish, apply a marker (a point) to the static image. It is important to note that the example below targets devices for S60 series.

Example

The simple idea is: (1) get the GPS position and (2) pass the latitude and longitude coordinates to the Web Services REST (url) of the Google Maps, (3) get the server response image in byte array type. So, our first step is shown below: The algorithm below must be implemented in a different thread. Otherwise, user interface may be blocked until the response comes back.

```
Criteria cr = new Criteria();
cr.setHorizontalAccuracy(500);
cr.setPreferredPowerConsumption(Criteria.NO_REQUIREMENT);

LocationProvider provider = null;
Location location = null;

try {
    provider = LocationProvider.getInstance(cr);
    location = provider.getLocation(120);
    provider.setLocationListener(this, -1, 0, 0);
} catch (LocationException e) {
    alert = new Alert("Error", "Location API Error", null, AlertType.ERROR);
} catch (InterruptedException e) {
    alert = new Alert("Error", "Thread Error", null, AlertType.ERROR);
}

Coordinates c = location.getQualifiedCoordinates();
if(c != null) {
    this.lat = c.getLatitude();
    this.lon = c.getLongitude();
}
```

Mini_App_With_Location_API_and_Google_Maps_in_Java_ME

In the beginning of the sample code above, we first set some criteria. Criteria fields include: accuracy, response time, need for altitude, and speed. This is up to the application to set these values. These criteria fields will filter the LocationProvider that meets the given values. After obtained a LocationProvider object, the application can use this object to obtain the location, in either of two ways:

- * Invoke a method synchronously to get a single location.
- * Register a listener and get periodic updates at application-defined intervals.

The location results can be got with the Location class. An instance of this class can available some critical informations, such as: coordinates, speed and textual address (if available), and a time stamp that indicates when the location measurements were made.

Coordinates can be represented with two different classes:

- * Coordinates: a point's latitude and longitude in degrees, and altitude in meters.
- * QualifiedCoordinates: latitude, longitude, and altitude, and also an indication of their accuracy.

Finally, we have our main information: latitude (through c.getLatitude() method) and longitude (through c.getLongitude() method). All we have to do now is open a HTTP connection with the Google Maps server.

```
HttpConnection httpConnection = null;
InputStream inputStream = null;
byte[] buffer = null;

try {
    httpConnection = (HttpURLConnection) Connector.open(url);
    httpConnection.setRequestMethod(HttpURLConnection.GET);
    inputStream = httpConnection.openInputStream();
    int length = (int)httpConnection.getLength();
    if(length > 0) {
        buffer = new byte[length];
        read(buffer);
    } else {
        int c;
        ByteArrayOutputStream byteArray = new ByteArrayOutputStream();
        while ((c = inputStream.read()) != -1){
            write(c);
        }
        buffer = byteArray.toByteArray();
        close(byteArray);
    }
} catch (Exception e) {
    = new AlertDialog.Builder(context).setTitle("Error").setMessage("Connection Failed").setPositiveButton("OK", null).create();
} finally {
    try {
        if(inputStream != null)
            inputStream.close();
        if(httpConnection != null)
            close();
    }
} catch (Exception e2) {
    = new AlertDialog.Builder(context).setTitle("Error").setMessage("IO Failed").setPositiveButton("OK", null).create();
}

return buffer;
```

Mini_App_With_Location_API_and_Google_Maps_in_Java_ME

The above code opens a http connection with a given URL and gets the server's response. This URL is a Web Service REST available by Google Maps, and is as follows:

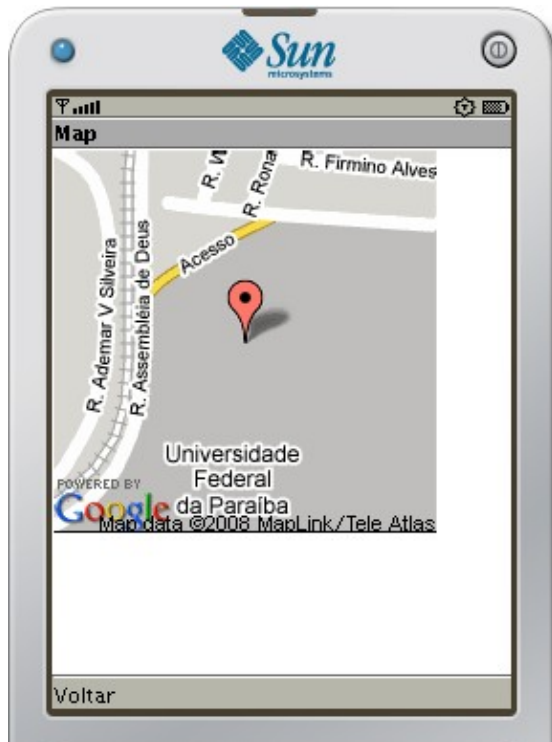
```
http://maps.google.com/staticmap?center=lat,lon&zoom=16&size=200x200&maptype=mobile&a
```

In this URL, we set some parameters for the static image: latitude and longitude, its zoom and its size (200x200), the map type (for mobile application) and a marker plotted based on lat and lon values. In our example, the application got (through GPS positioning) the Federal University of Campina Grande latitude and longitude values.

The image returned from the web services (byte[]) is used to print the map in the screen.

```
Form f = new Form("Map");  
f.append(Image.createImage(buffer, 0, buffer.length))
```

The result is:



For more application examples

See <http://efforts.embedded.ufcg.edu.br/javame>