Location Based Services

Introduction

- The mapping API and the location API are isolated from each other.
- The mapping API is not part of the Android project. It is a separated API developed by Google.
- We can use either the API for native development (v2) or the JavaScript API and develop an hybrid application.

Google Maps Android API v1

The Map Key

- In order to interact with the google map service we first need to obtain a map key. We need to get two separated keys. One for development. The other for production.
- In order to get the map key from google we first need to get the MD-5 digital signature we use to sign our application.

The MD-5 Signature

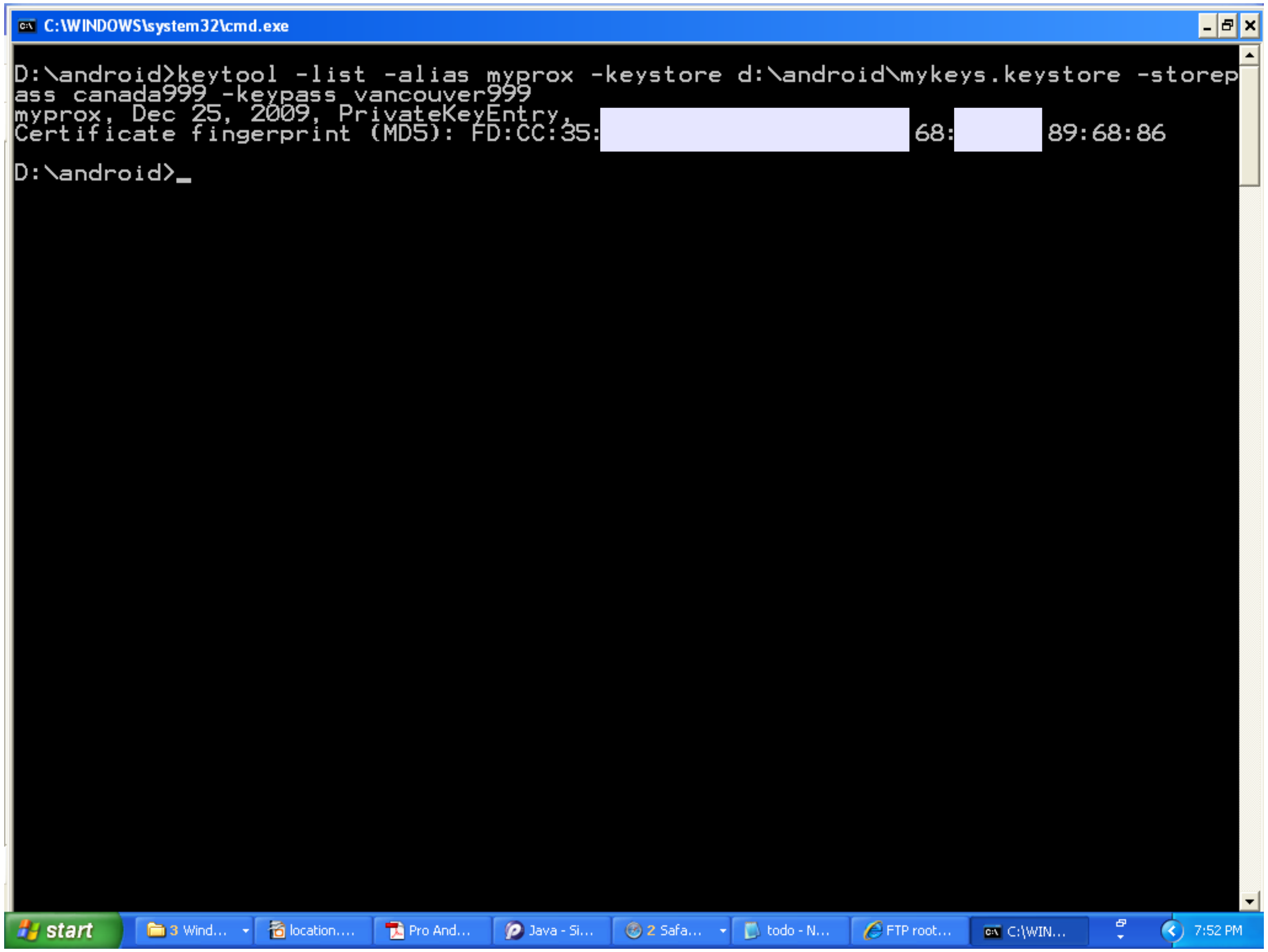
- We can get the MD-5 digital signature by calling the keytool utility on our keystore file passing over '-list' option.
- When dealing with the debug signature we can find the location of the keystore file browsing at `Windows->Preferences->Android->Build`.

The MD-5 Signature

❖ We should call the keytool utility in the following way:

```
keytool -list    -alias our_application_signature_alias
                 -keystore "e:\android\temp\mykeys.keystore"
                 -storepass mykeys_password
                 -keypass mykeys_entrance_password
```

The MD-5 Signature



```
D:\android>keytool -list -alias myprox -keystore d:\android\mykeys.keystore -storep
ass canada999 -keypass vancouver999
myprox, Dec 25, 2009, PrivateKeyEntry,
Certificate fingerprint (MD5): FD:CC:35:[redacted]68:[redacted]89:68:86

D:\android>_
```

The MD-5 Signature

- In order to get the MD5 digital signature of the Debug Certificate we should execute the following code:

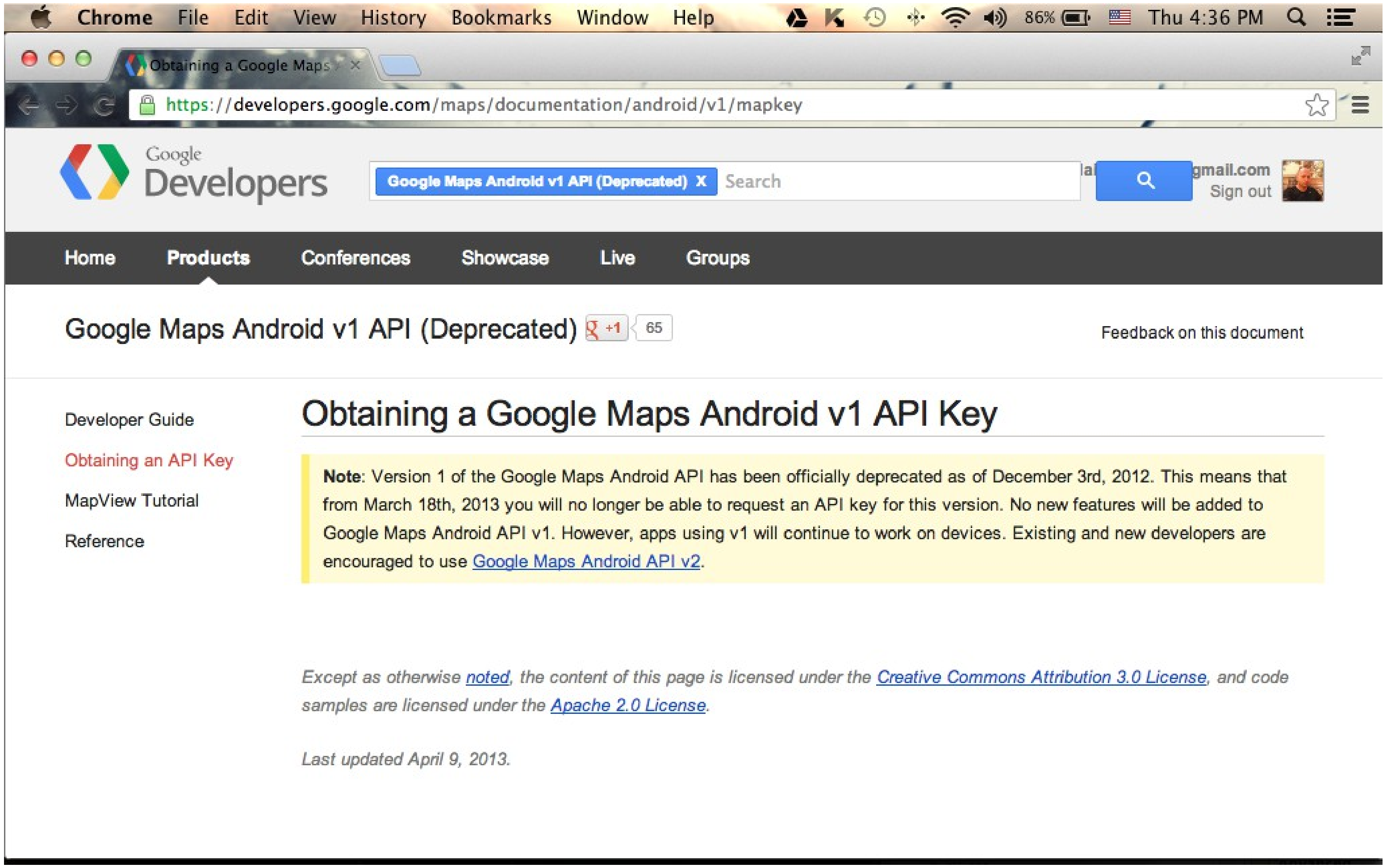
  ```
  keytool -list -alias androiddebugkey -keystore
  "C:\Documents and Settings\sh\Local Settings\Application
  Data\Android\debug.keystore"
  -storepass android -keypass android
  ```

  This is assuming the keystore file is indeed located in the specified directory. If the debug keystore file is located in another folder you just need to update this code with the new location.

Google Maps API v1 Key

- Now, that we have the MD5 digital signature we can browse google maps web site and get the required key.
- Getting the key for Google Maps API v1 at the following URL address is no longer feasible. As of March 18th 2013 we can no long get a key for using Google Maps API v1 at http://code.google.com/android/maps-api-signup.html.

Google Maps API v1 Key


Chrome File Edit View History Bookmarks Window Help
86% Thu 4:36 PM
Obtaining a Google Maps
https://developers.google.com/maps/documentation/android/v1/mapkey
Google Developers
Google Maps Android v1 API (Deprecated) X
Search
gmail.com
Sign out
Home
Products
Conferences
Showcase
Live
Groups
+1
65
Feedback on this document
Developer Guide
Obtaining an API Key
MapView Tutorial
Reference

Google Maps Android v1 API (Deprecated)

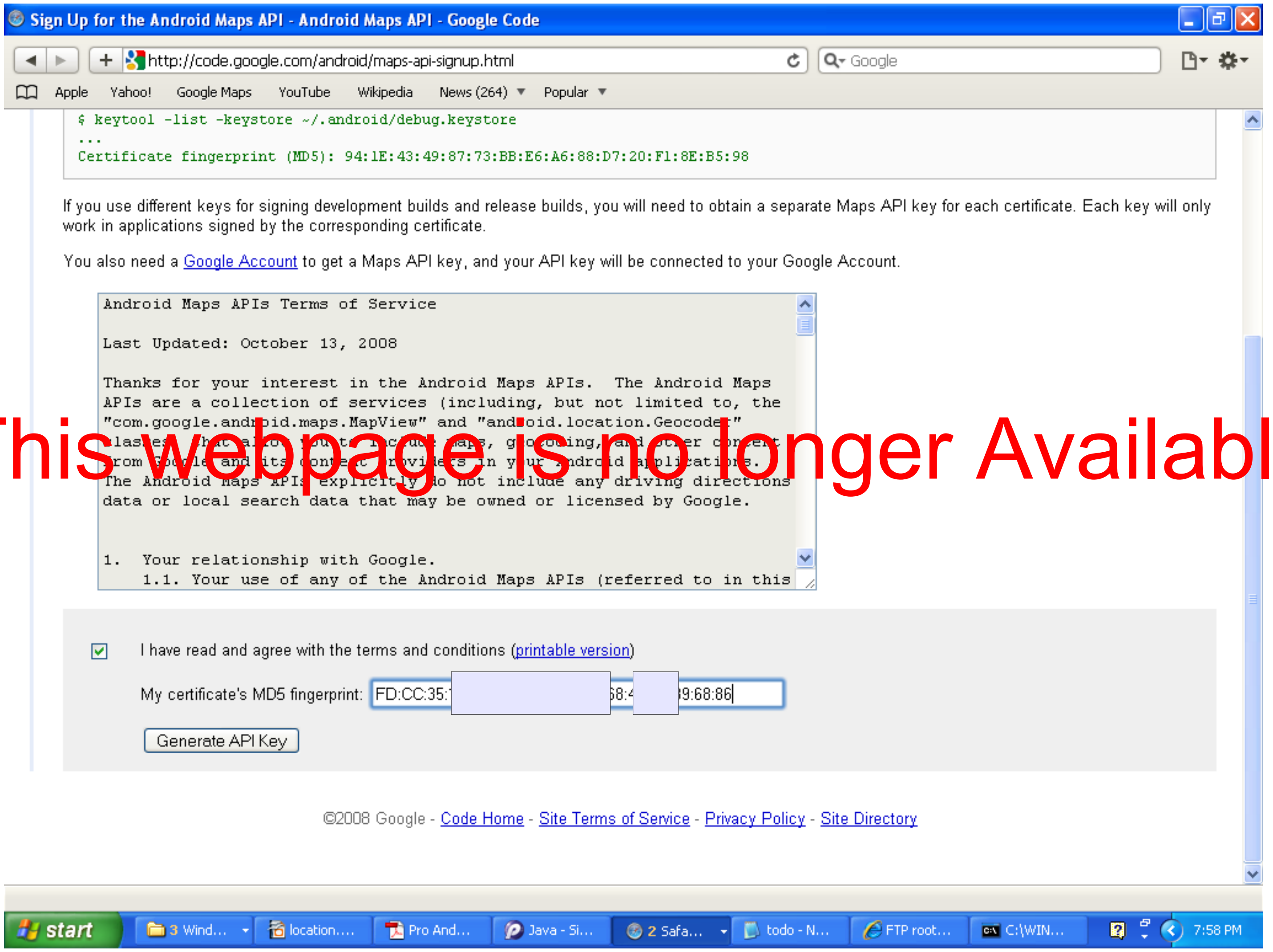
Obtaining a Google Maps Android v1 API Key

Note: Version 1 of the Google Maps Android API has been officially deprecated as of December 3rd, 2012. This means that from March 18th, 2013 you will no longer be able to request an API key for this version. No new features will be added to Google Maps Android API v1. However, apps using v1 will continue to work on devices. Existing and new developers are encouraged to use Google Maps Android API v2.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License.

Last updated April 9, 2013.

Google Maps API v1 Key



Sign Up for the Android Maps API - Android Maps API - Google Code

http://code.google.com/android/maps-api-signup.html

```
$ keytool -list -keystore ~/.android/debug.keystore
...
Certificate fingerprint (MD5): 94:1E:43:49:87:73:BB:E6:A6:88:D7:20:F1:8E:B5:98
```

If you use different keys for signing development builds and release builds, you will need to obtain a separate Maps API key for each certificate. Each key will only work in applications signed by the corresponding certificate.

You also need a Google Account to get a Maps API key, and your API key will be connected to your Google Account.

```
Android Maps APIs Terms of Service

Last Updated: October 13, 2008

Thanks for your interest in the Android Maps APIs.  The Android Maps
APIs are a collection of services (including, but not limited to, the
"com.google.android.maps.MapView" and "android.location.Geocoder"
classes that allow you to include maps, geocoding, and other content
from Google and its content providers in your Android applications.
The Android Maps APIs explicitly do not include any driving directions
data or local search data that may be owned or licensed by Google.


1.  Your relationship with Google.
    1.1. Your use of any of the Android Maps APIs (referred to in this
```

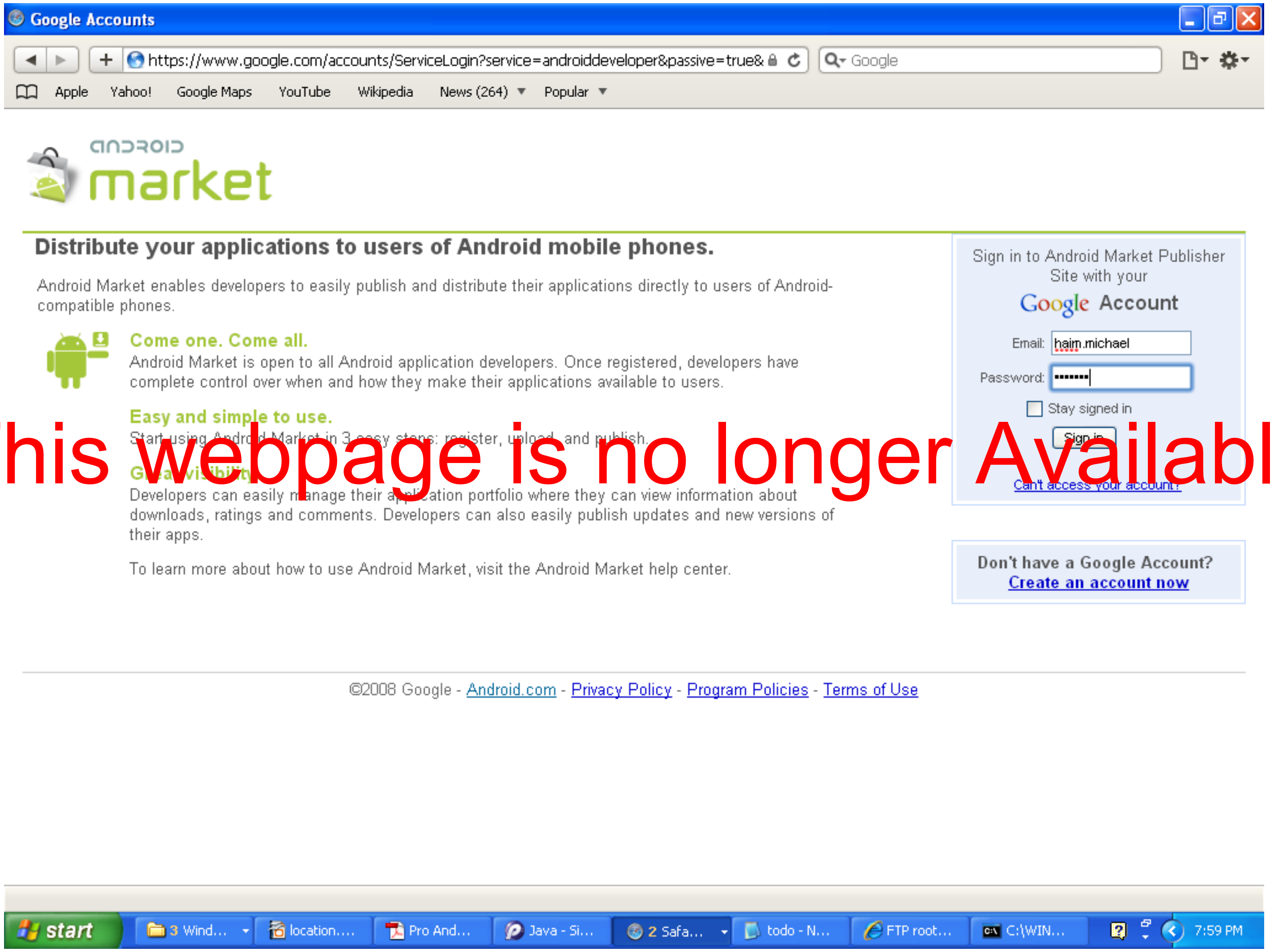
[✓] I have read and agree with the terms and conditions (printable version)

My certificate's MD5 fingerprint: FD:CC:35:1[redacted]8:4[redacted]9:68:86

©2008 Google - Code Home - Site Terms of Service - Privacy Policy - Site Directory

This webpage is no longer Available!

Google Maps API v1 Key



Google Accounts

https://www.google.com/accounts/ServiceLogin?service=androiddeveloper&passive=true&

android market

Distribute your applications to users of Android mobile phones.

Android Market enables developers to easily publish and distribute their applications directly to users of Android-compatible phones.

Come one. Come all.
Android Market is open to all Android application developers. Once registered, developers have complete control over when and how they make their applications available to users.

Easy and simple to use.
Start using Android Market in 3 easy steps: register, upload, and publish.

Great visibility.
Developers can easily manage their application portfolio where they can view information about downloads, ratings and comments. Developers can also easily publish updates and new versions of their apps.

To learn more about how to use Android Market, visit the Android Market help center.

Sign in to Android Market Publisher Site with your Google Account

Email: haim.michael

Password: ••••••••

Stay signed in

Sign in

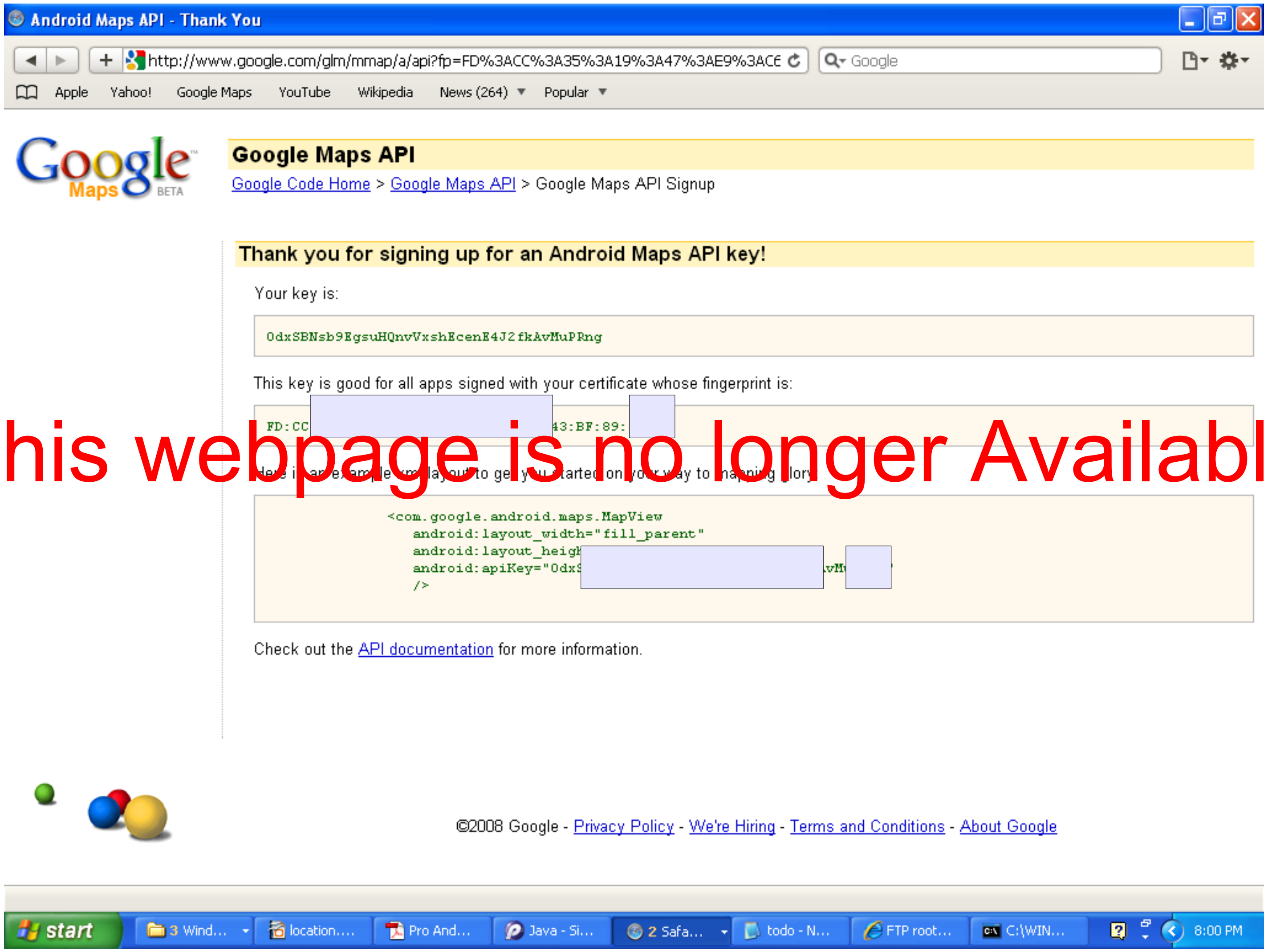
Can't access your account?

Don't have a Google Account?
Create an account now

©2008 Google - Android.com - Privacy Policy - Program Policies - Terms of Use

This webpage is no longer Available!

Google Maps API v1 Key



Android Maps API - Thank You

http://www.google.com/glm/mmap/a/api?fp=FD%3ACC%3A35%3A19%3A47%3AE9%3ACE

Apple Yahoo! Google Maps YouTube Wikipedia News (264) Popular

Google Maps API

Google Code Home > Google Maps API > Google Maps API Signup

Thank you for signing up for an Android Maps API key!

Your key is:

```
0dxSBNsb9EgsuHQnvVxshEcenE4J2fkAvMuPRng
```

This key is good for all apps signed with your certificate whose fingerprint is:

```
FD:CC[redacted]43:BF:89:[redacted]
```

This webpage is no longer Available!

Here is an example xml layout to get you started on your way to mapping glory:

```
<com.google.android.maps.MapView
   android:layout_width="fill_parent"
   android:layout_heigh[redacted]
   android:apiKey="0dx[redacted]vM[redacted]
   />
```

Check out the API documentation for more information.

©2008 Google - Privacy Policy - We're Hiring - Terms and Conditions - About Google

start 3 Wind... location.... Pro And... Java - Si... 2 Safa... todo - N... FTP root... C:\WIN... 8:00 PM

Google Maps API v1 Key

- Once we have the key we can start using the `MapView` user interface control.
- Although it is no longer possible to get the key for using Google Maps for Android v1, applications that already have a key can continue using it.

The `<uses-library>` XML Element

- When using classes that are not part of the android platform as in the case of using classes that belong to Google Maps API as in the case with using Google Maps API v1 dealing with the com.google.android.maps package we should add the `<uses-library>` XML element into the `AndroidManifest.xml` file.

```
<uses-library android:name="com.google.android.maps"/>
```

Google Maps API v1 Sample

```
package com.abelski.android.samples;

import com.google.android.maps.MapActivity;
import android.os.Bundle;


public class JWorldViewActivity extends MapActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected boolean isRouteDisplayed()
    {
        // TODO Auto-generated method stub
        return false;
    }
}
```

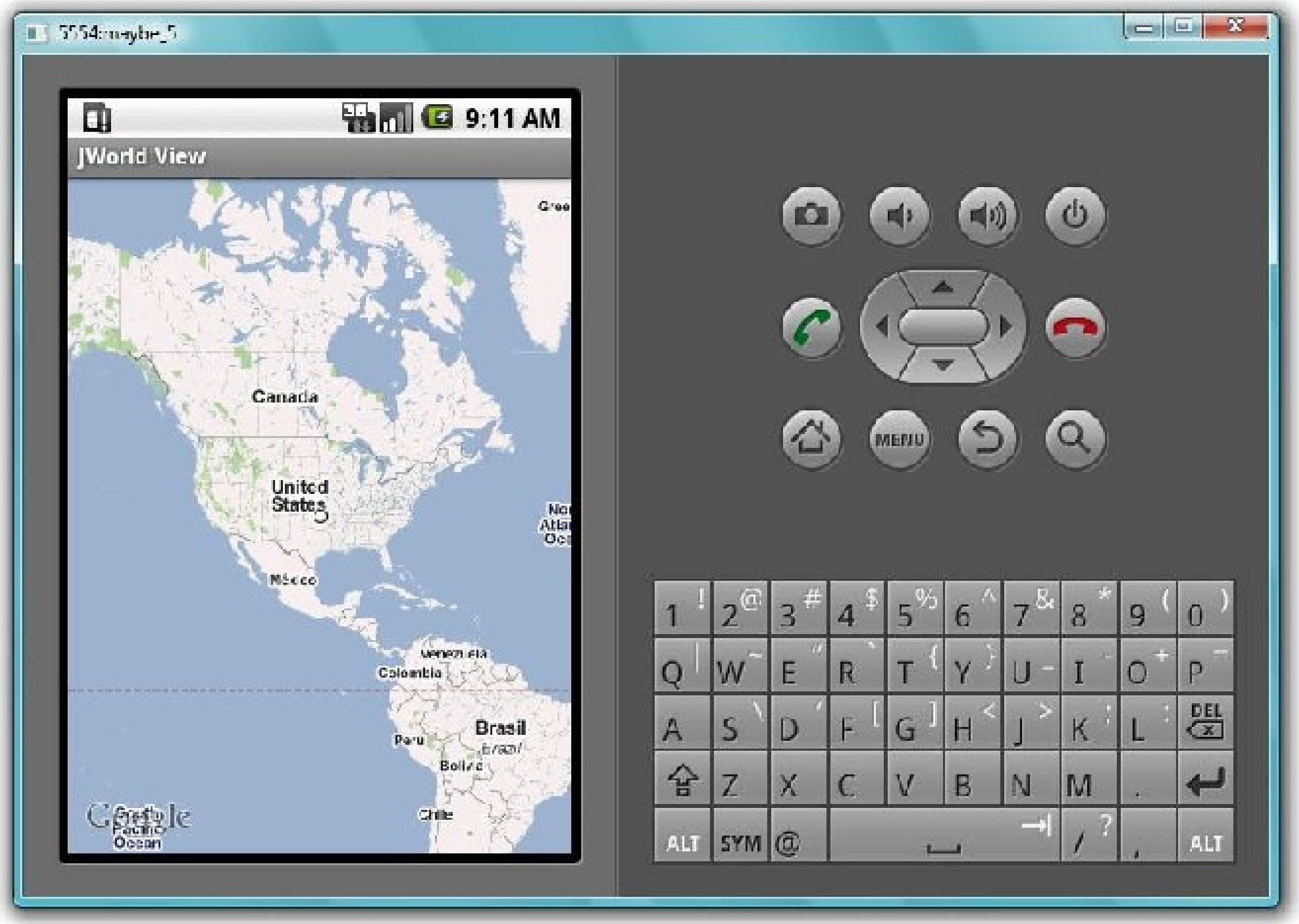
Google Maps API v1 Sample

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <view class="com.google.android.maps.MapView"
        android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:apiKey="___enter_your_key___"
        />

</RelativeLayout>
```

Google Maps API v1 Sample

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.abelski.android.samples"
      android:versionCode="1"
      android:versionName="1.0">

<application android:icon="@drawable/icon"
  android:label="@string/app_name">
<activity android:name=".JWorldViewActivity"
  android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-library android:name="com.google.android.maps"/>
</application>

<uses-sdk android:minSdkVersion="4" />
<uses-permission android:name="android.permission.INTERNET"/>
<user-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<user-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
</manifest>
```

Sample


5554:maybe_5
9:11 AM
JWorld View
Canada
United States
México
Venezuela
Colombia
Brasil
Peru
Bolivia
Chile
Google
MENU

The Map Controller

- Each map view has a controller. The controller is a `MapController` object.
- We can get it by calling the `getController()` method on the `MapView` object we are working with.
- The `MapController` class defines several useful methods we can execute on the `MapController` object we are working with.

The Map Controller

`public void stopPanning()`

Resets the pan state to make the map stationary. This could be necessary if we receive a key-down event but will never receive the corresponding key-up.

```
public boolean onKey(  android.view.View v,
                       int keyCode,android.view.KeyEvent event)
```

Processes key events and translates them into appropriate panning of the map. Defined in `View.OnKeyListener`.

`public void animateTo(GeoPoint point)`

Start animating the map towards the given point.

The Map Controller

```
public void scrollBy(int x, int y)
```

Scroll by a given amount, in pixels. The scrolling won't be involved with animation.

```
public void setCenter(GeoPoint point)
```

Set the map view to the given center. There will be no animation.

```
public void stopAnimation(boolean jumpToFinish)
```

Stops any animation that may be in progress, and conditionally update the map center to whatever offset the partial animation had achieved. If the passed value is true, we'll shortcut the animation to its endpoint. if false, we'll cut it off where it stands.

```
public int setZoom(int zoomLevel)
```

Sets the zoomlevel of the map. The value will be clamped to be between 1 and 21 inclusive.

The Map Controller

```
public boolean zoomIn()
```

Zoom in by one zoom level. This begins an animated zoom step.

```
public boolean zoomOut()
```

Zoom out by one zoom level. This begins an animated zoom step.

```
public boolean zoomInFixing(int xPixel, int yPixel)
```

Zoom in by one zoom level. This begins an animated zoom step. xPixel is the offset, in pixels from the left of the map, where the fixed point of our zoom will be. yPixel is the offset, in pixels from the top of the map, where the fixed point of our zoom will be.

```
public boolean zoomOutFixing(int xPixel, int yPixel)
```

Zoom out by one zoom level. This begins an animated zoom step. xPixel is the offset, in pixels from the left of the map, where the fixed point of our zoom will be. yPixel is the offset, in pixels from the top of the map, where the fixed point of our zoom will be.

The Map Controller

```
public void zoomToSpan(int latSpanE6, int lonSpanE6)
```

Attempts to adjust the zoom of the map so that the given span of latitude and longitude will be displayed.

```
public void animateTo(GeoPoint point, java.lang.Runnable runnable)
```

Start animating the map towards the given point. If and when the animation reaches its natural conclusion, this callback will be run on the UI thread. The callback will not be run if the animation is aborted.

```
public void animateTo(GeoPoint point, android.os.Message message)
```

Start animating the map towards the given point. If and when the animation reaches its natural conclusion, dispatch the given message (if non-null). The message will not be dispatched if the animation is aborted.

Maps Overlays

- On top of the map we can place custom data in the form of pushpins or small balloon markers that indicate specific locations.
- Customized data we want to add on top of our map is represented by an `Overlay` object.

Maps Overlays

- The `Overlay` class is an abstract one. We can work with objects instantiated from a class that extends the `ItemizedOverlay` class. It is another abstract class that already extends `Overlay` and includes the definitions for some of the methods.
- Each specific location on our map is represented by a `GeoPoint` object. The location is represented by its latitude and longitude, in micro degrees.

Overlays Google Maps API v1 Sample

```
package com.abelski.android.samples;

import com.google.android.maps.MapActivity;

import android.graphics.drawable.Drawable;
import android.os.Bundle;
import com.google.android.maps.MapView;
import android.widget.LinearLayout;


public class JWorldViewActivity extends MapActivity
{


    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        MapView map = (MapView)findViewById(R.id.map);
        map.setBuiltInZoomControls(true);
        map.setClickable(true);
```

Overlays Google Maps API v1 Sample

```
        Drawable mapMarker =
            getResources().getDrawable(R.drawable.my_marker);

        mapMarker.setBounds(0,0,
            mapMarker.getIntrinsicWidth(),
            mapMarker.getIntrinsicHeight());

        map.getOverlays().add(new MyLocations(mapMarker));
    }

    @Override
    protected boolean isRouteDisplayed()
    {
        // TODO Auto-generated method stub
        return false;
    }
}
```

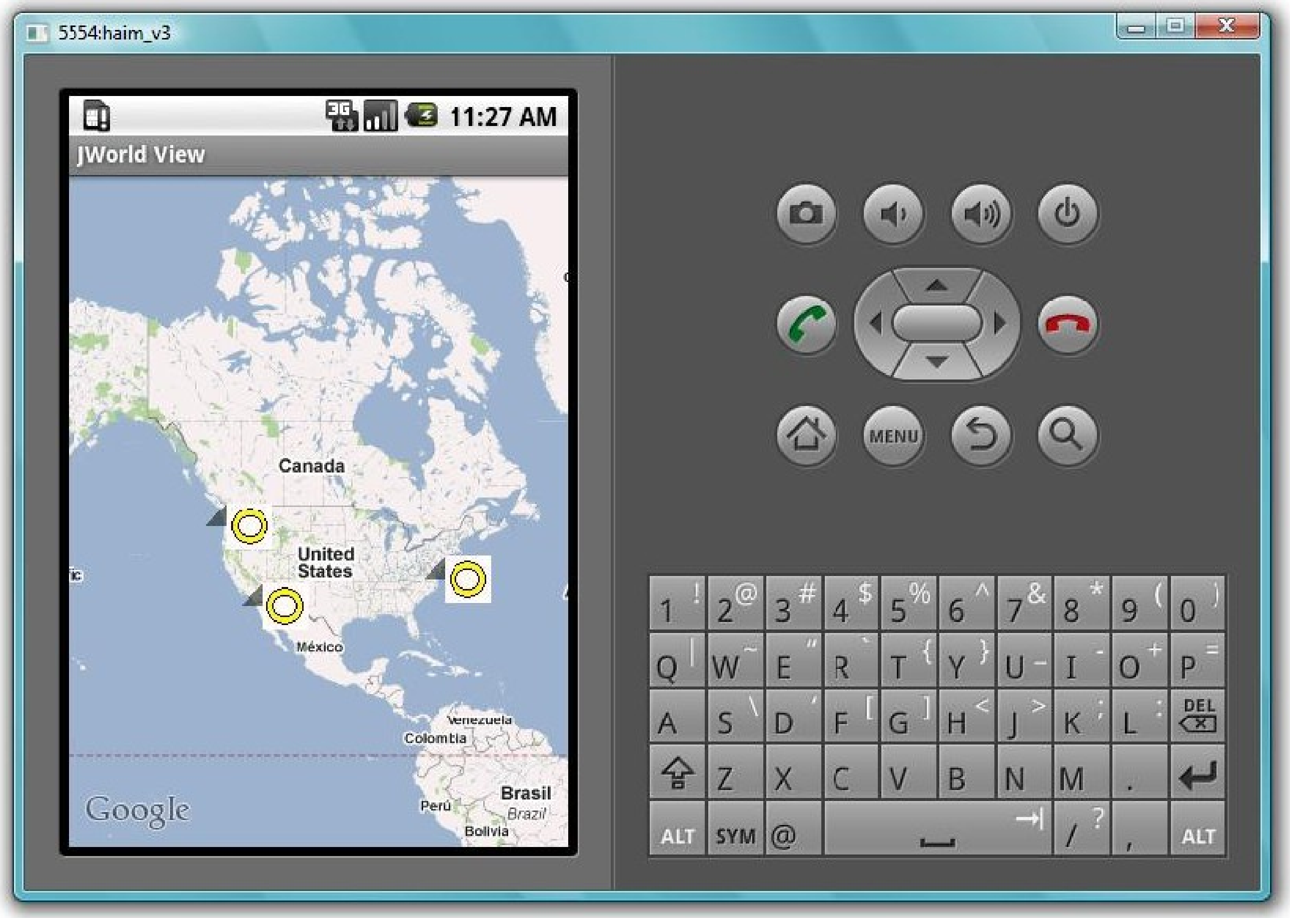
Overlays Google Maps API v1 Sample

```
package com.abelski.android.samples;

import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;
import com.google.android.maps.GeoPoint;
import java.util.ArrayList;
import java.util.List;
import android.graphics.drawable.Drawable;


public class MyLocations extends ItemizedOverlay
{
    private List<OverlayItem> locations = new ArrayList<OverlayItem>();
    private Drawable marker;

    public MyLocations(Drawable markerOnMap)
    {
        super(markerOnMap);
        this.marker = markerOnMap;
        GeoPoint ubcPitPub = new GeoPoint((int)(49.267446*1000000),
            (int)(-123.250414*1000000));
        GeoPoint vegasCity = new GeoPoint((int)(36.188875*1000000),
            (int)(-115.051575*1000000));
        GeoPoint newYork = new GeoPoint((int)(40.753499*1000000),
            (int)(-73.927002*1000000));
```

Overlays Google Maps API v1 Sample

```
            locations.add(new OverlayItem(ubcPitPub,"UBC",
                "University of British Columbia Students Club"));
            locations.add(new OverlayItem(vegasCity,"Las Vegas",
                "Las Vegas City View"));
            locations.add(new OverlayItem(newYork,"NYC","New York City"));
            populate();
        }

        @Override
        protected OverlayItem createItem(int i)
        {
            return locations.get(i);
        }

        @Override
        public int size()
        {
            return locations.size();
        }
    }
```

Overlays Google Maps API v1 Sample



Location Based Services Permissions

Location Based Services Permissions

- When using the location based services there is a need in the following permissions (at the minimum):

```
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
```

This permission is required in order to get data from the GPS.

```
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

This permission is required in order to get data from the wifi connectivity.

```
<uses-permission android:name="android.permission.INTERNET" />
```

This permission is required in order to access the internet.

The `Geocoder` Class

- Geocoding is the process of translating a an address or a location into a pair of latitude and longitude numbers.
- The `location.Geocoder` class provides this service. Using this class we can translate in both directions. It can take an address and returns a pair of latitude and longitude numbers and it can take a pair of latitude and longitude numbers and return a list of addresses.

The `Geocoder` Class

```
public List<Address> getFromLocation (
          double latitude,
          double longitude,
          int maxResults)

public List<Address> getFromLocationName (
          String locationName,
          int maxResults,
          double lowerLeftLatitude,
          double lowerLeftLongitude,
          double upperRightLatitude,
          double upperRightLongitude)

public List<Address> getFromLocationName (
          String locationName,
          int maxResults)
```

The `Address` Class

- The Address class describes a physical location using a set of strings in accordance with the xAL (eXtensible Address Language) as described at the following specification http://www.oasis-open.org/committees/ciq/ciq.html#6.

The `LocationManager` Class

- This `android.location.LocationManager` class provides us with two mechanisms.
- The first is the ability to get the device geographical location.
- The other is the ability to be notified (via an intent) when the device enters a predefined geographical location.

The `LocationManager` Class

❖ We get a `LocationManager` object by calling the `getSystemService` method that was defined in `Activity`.

```
LocationManager manager = (LocationManager)this.
    getSystemService(Context.LOCATION_SERVICE);
```

The `Location` Class

- Calling the `getLastKnownLocation` method on our `LocationManager` object we should get a `Location` object that describes our geographic location.
- Calling this method we should pass over the name of the location provider from which we want to get the information.

```
...
Location loc = manager.getLastKnownLocation(
    LocationManager.GPS_PROVIDER);
...
```

The `Location` Class

- Once we get a Location object there are various methods we can call on it in order to get geographic related information such as the following:

  The Device Atitude & Longtitude

  Most location providers are capable of providing this basic information.

  The Device Altitude

  Calling `hasAltitude()` we shall know whether the altitude information is available or not.

  The Device Bearing

  This method returns the degrees east of the true north. Calling `hasBearing()` we shall know whether this information is available or not.

The `Location` Class

The Device Speed

Calling `hasSpeed()` we can know whether the speed information is available or not.

The `LocationProvider` Class

- The `LocationManager` class provides access to the available location providers (services).
- Each `LocationProvider` object represents a specific location service available on our handset.

The `LocationProvider` Class

- Calling the `getAllProviders()` method on our `LocationManager` object we shall get a `List` object that holds the names of the available location providers.

```
...
List<String> providerList = manager.getAllProviders();
...
```

The `LocationProvider` Class

- Calling the `getProvider()` method on our `LocationManager` object we should passover the name of the requested content provider. In return we shall get a `LocationProvider` object that represents the specific location provider we passed over its name.

```
...
LocationProvider provider = manager.getProvider(String name);
...
```

The `LocationProvider` Class

- Calling the `getBestProvider()` method on our `LocationManager` object we shall get a reference for a `LocationProvider` object that represents the best matching available location provider.

The `LocationProvider` Class

- Calling the `getBestProvider()` we should passover a reference for a `Criteria` object that describes the required characteristics of the location provider we need.

```
...
Criteria criteria = new Criteria();
criteria.setAltitudeRequired(true);
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setCostAllowed(true);
...
LocationProvider provider = manager.getBestProvider(criteria);
...
```

The `LocationListener` Interface

- Implementing this interface we shall get a class that its objects can be used as listeners for location changes.
- Calling the `requestLocationUpdates()` method on our location manager passing over a `LocationListener` object will tie between the two.
- Each time a location update is received the `onLocationChanged` method will be called on the `LocationListener` object.

The `LocationListener` Interface

```
LocationManager manager =
        (LocationManager)getSystemService(Context.LOCATION_SERVICE);

LocationListener listener = new LocationListener()
    {
        public void onLocationChanged(Location location)
        {
            if (location != null)
            {
                Toast.makeText(getBaseContext(),
                        "[" + location.getLatitude() +
                        "] [" + location.getLongitude() + "]",
                        Toast.LENGTH_SHORT).show();
            }
        }
```

The `LocationListener` Interface

```
        public void onProviderDisabled(String provider)
        {
            ...
        }
        public void onProviderEnabled(String provider)
        {
            ...
        }
        public void onStatusChanged(String provider, int status,
            Bundle extras)
        {
            ...
        }
    };

manager.requestLocationUpdates(manager.GPS_PROVIDER,0,0,listener);
```

The `LocationListener` Interface

- When we no longer need the location updates we can call the `removeUpdates()` method passing over the reference for the `LocationListener` object we registered.

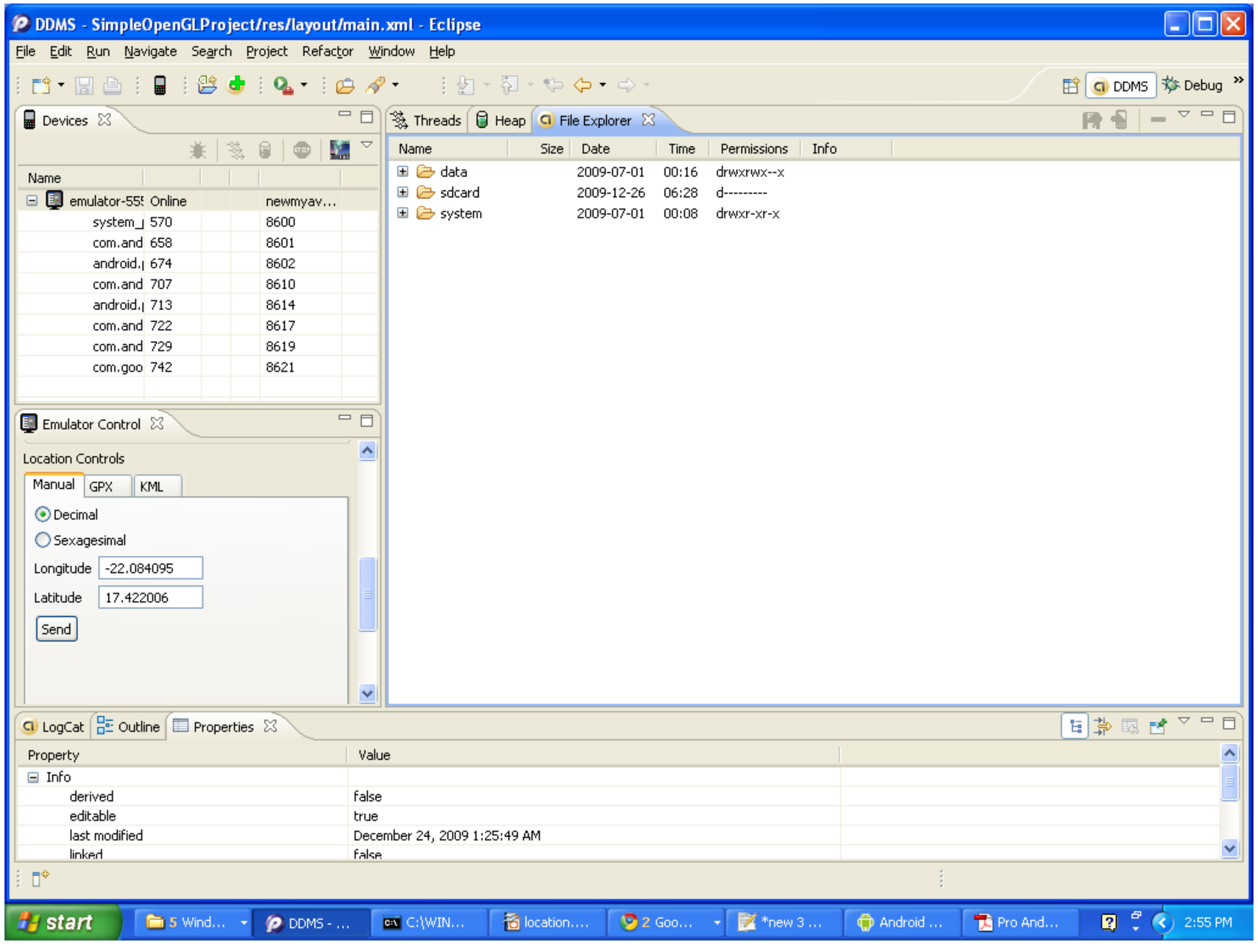
```
manager.removeUpdates(listener);
```

- If we avoid calling this method the application will continue to receive location updates even after the relevant activity is closed, which will eventually drain the battery.

The Debug Monitor Service

- The android eclipse plug-in includes the DDMS (Debug Monitor Service).
- We can use it to pass over the emulator information about a new location.

The Debug Monitor Service



Name			
emulator-55!	Online		newmyav...
system_	570		8600
com.and	658		8601
android.	674		8602
com.and	707		8610
android.	713		8614
com.and	722		8617
com.and	729		8619
com.goo	742		8621

Name	Size	Date	Time	Permissions	Info
data		2009-07-01	00:16	drwxrwx--x	
sdcard		2009-12-26	06:28	d---------	
system		2009-07-01	00:08	drwxr-xr-x	

Location Controls

Manual | GPX | KML

(●) Decimal
( ) Sexagesimal

Longitude -22.084095
Latitude 17.422006

Send

Property	Value
Info	
derived	false
editable	true
last modified	December 24, 2009 1:25:49 AM
linked	false

© 2008 Haim Michael

Proximity Alerts

- One of the methods we can call on a `LocationManager` object is the `addProximityAlert`(). This method allows us to register a `PendingIntent` object that will be fired when the device gets within a certain distance of a certain location.

Proximity Alerts

- Calling the `addProximityAlert()` method we should pass over five arguments: `latitude`, `longitude`, `radius` (meters), `expiration` (milliseconds) and a reference for a `PendingIntent` object that will be fired when the device detects that it has entered or exited the area surrounding the location.

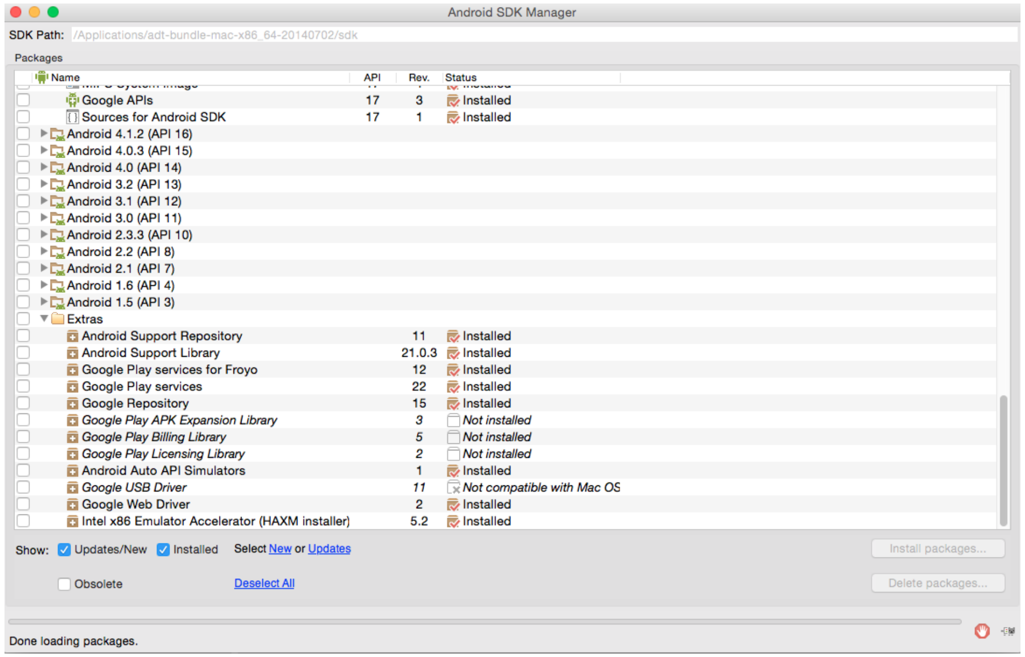
Google Maps Android API Version 2

Introduction

- The new Google Maps API provides us with many new exciting features such as 3D maps, indoor mapping and vector-based tiles for efficient caching and drawing. The new Google Maps API requires API level 12 or higher.
- The following slides overview the steps required for using the Google Maps Android API v2 when developing our project using the Android Studio.

The Google Play Services

❖ Use the Android SDK manager to ensure that the Google Play Services is installed. The Google Maps Android API v2 is part of the Google Play Services.



Android SDK Manager

SDK Path: /Applications/adt-bundle-mac-x86_64-20140702/sdk

Packages

Name	API	Rev.	Status
Google APIs	17	3	Installed
Sources for Android SDK	17	1	Installed
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 4.0 (API 14)			
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Android 2.1 (API 7)			
Android 1.6 (API 4)			
Android 1.5 (API 3)			
Extras			
Android Support Repository		11	Installed
Android Support Library		21.0.3	Installed
Google Play services for Froyo		12	Installed
Google Play services		22	Installed
Google Repository		15	Installed
Google Play APK Expansion Library		*3*	*Not installed*
Google Play Billing Library		*5*	*Not installed*
Google Play Licensing Library		*2*	*Not installed*
Android Auto API Simulators		1	Installed
Google USB Driver		*11*	*Not compatible with Mac OS*
Google Web Driver		2	Installed
Intel x86 Emulator Accelerator (HAXM installer)		5.2	Installed

Show: [✓ Updates/New] [✓ Installed] Obsolete

Select New or Updates

Deselect All

Done loading packages.

The Google Play Services

- In order to make the Google Play Services API available for the project we develop we should update the `build.gradle` file. We should add a new build rule to the dependencies section of our project. It should be a rule that refers the latest version of google play services.

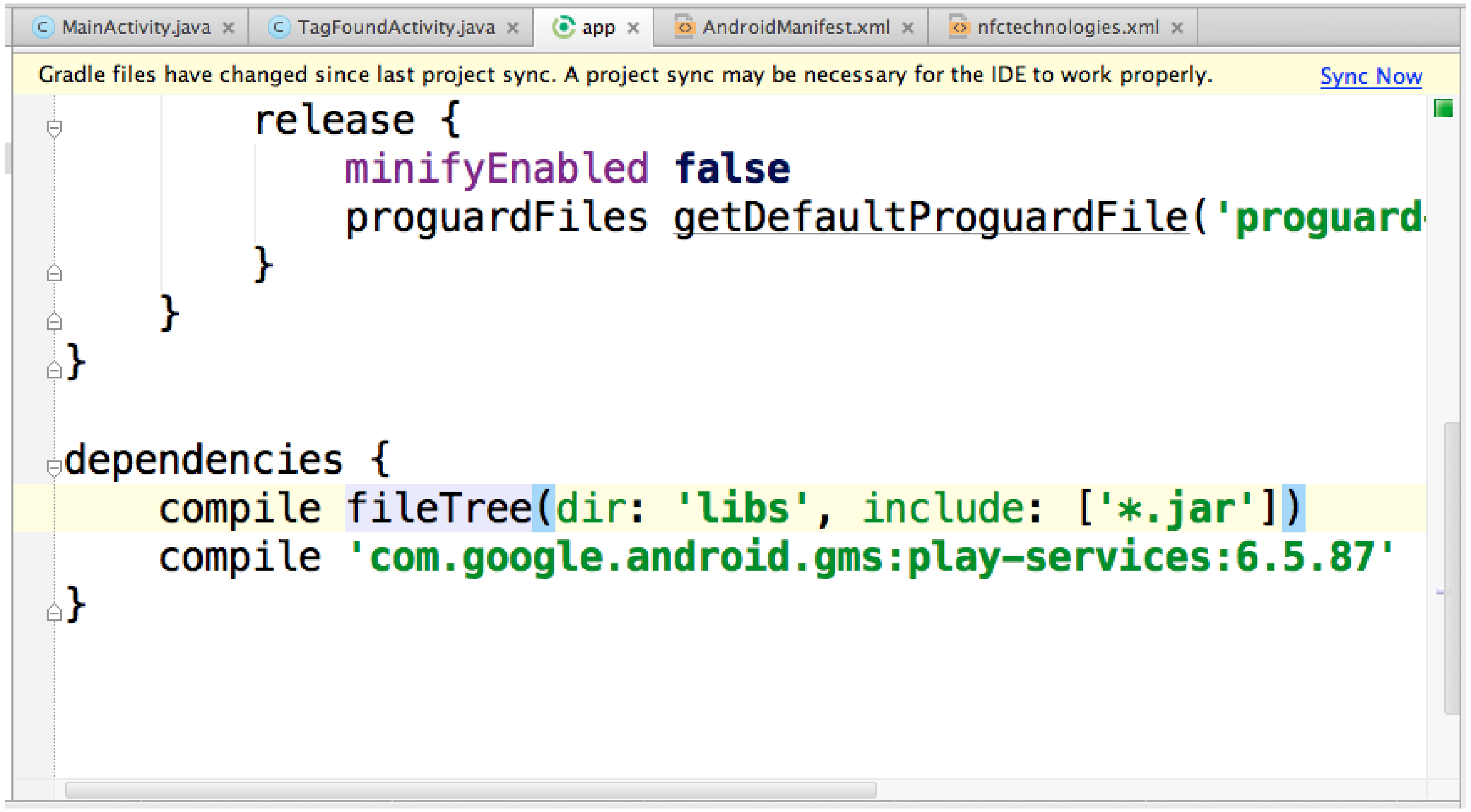
The Google Play Services

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    ...
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.gms:play-services:6.5.87'
}
```

The Google Play Services

- Whenever a new Google Play Services version is released we better update the `build.gradle` file with the new version.
- Once the build.gradle was is updated we should sync the project with it.

The Google Play Services



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.gms:play-services:6.5.87'
}
```

The Google Play Services

- We should now add the following meta data element as a new child of the application element in our manifest file. This element is required in order to inform that our application uses the google play services.

```
<meta-data android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

The Google Play Services

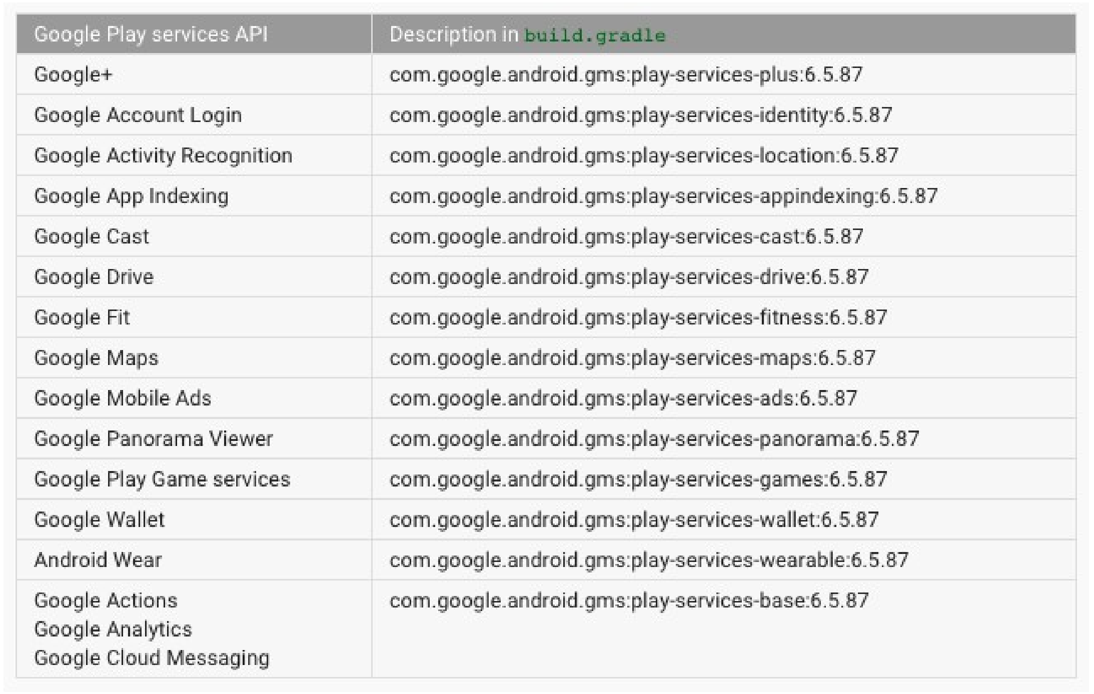
- We should now update the dependencies section in our project build.gradle file with specific compile instructions for adding those specific parts of the google play services that we need.
- In order to use Google Maps we should add the following:

```
compile 'com.google.android.gms:play-services-maps:6.5.87'
```

The Google Play Services

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    ...
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.gms:play-services-maps:6.5.87'
}
```

The Google Play Services



Google Play services API	Description in `build.gradle`
Google+	com.google.android.gms:play-services-plus:6.5.87
Google Account Login	com.google.android.gms:play-services-identity:6.5.87
Google Activity Recognition	com.google.android.gms:play-services-location:6.5.87
Google App Indexing	com.google.android.gms:play-services-appindexing:6.5.87
Google Cast	com.google.android.gms:play-services-cast:6.5.87
Google Drive	com.google.android.gms:play-services-drive:6.5.87
Google Fit	com.google.android.gms:play-services-fitness:6.5.87
Google Maps	com.google.android.gms:play-services-maps:6.5.87
Google Mobile Ads	com.google.android.gms:play-services-ads:6.5.87
Google Panorama Viewer	com.google.android.gms:play-services-panorama:6.5.87
Google Play Game services	com.google.android.gms:play-services-games:6.5.87
Google Wallet	com.google.android.gms:play-services-wallet:6.5.87
Android Wear	com.google.android.gms:play-services-wearable:6.5.87
Google Actions Google Analytics Google Cloud Messaging	com.google.android.gms:play-services-base:6.5.87

These are the available APIs we can add to the build.gradle file

Creating ProGuard Exception

- In order to prevent ProGuard from damaging classes we need we can add the following lines into the proguard-project.txt file.

```
-keep class * extends java.util.ListResourceBundle {
    protected Object[][] getContents();
}

-keep public class
com.google.android.gms.common.internal.safeparcel.SafeParcelable {
    public static final *** NULL;
}
```

Creating ProGuard Exception

```
-keepnames @com.google.android.gms.common.annotation.KeepName class *

-keepclassmembernames class * {

    @com.google.android.gms.common.annotation.KeepName *;

}


-keepnames class * implements android.os.Parcelable {

    public static final ** CREATOR;

}
```

Google Maps API Key

- In order to add the Google Maps API key to our application we should browse the Google APIs Console website. Make sure you have your project's SHA-1 fingerprint. You can get it using the keytool utility (part of the JDK).

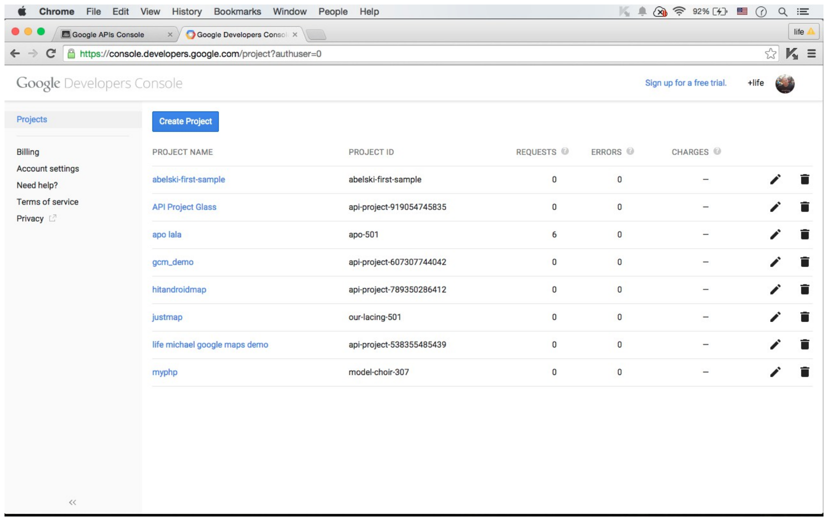
Google Maps API Key

- Using mac/linux, in order to get the SHA-1 fingerprint for the application (in debugging phase) you should use the following command:

```
keytool -list -v -keystore ~/.android/debug.keystore
    -alias androiddebugkey
    -storepass android
    -keypass android
```

Google Maps API Key

❖ You can now visit the Google APIs Console website and get a valid Google Maps API key for you can use in your project.


Chrome File Edit View History Bookmarks Window People Help
Google APIs Console
Google Developers Consol
https://console.developers.google.com/project?authuser=0
Google Developers Console
Sign up for a free trial.
+life
Projects
Billing
Account settings
Need help?
Terms of service
Privacy
Create Project

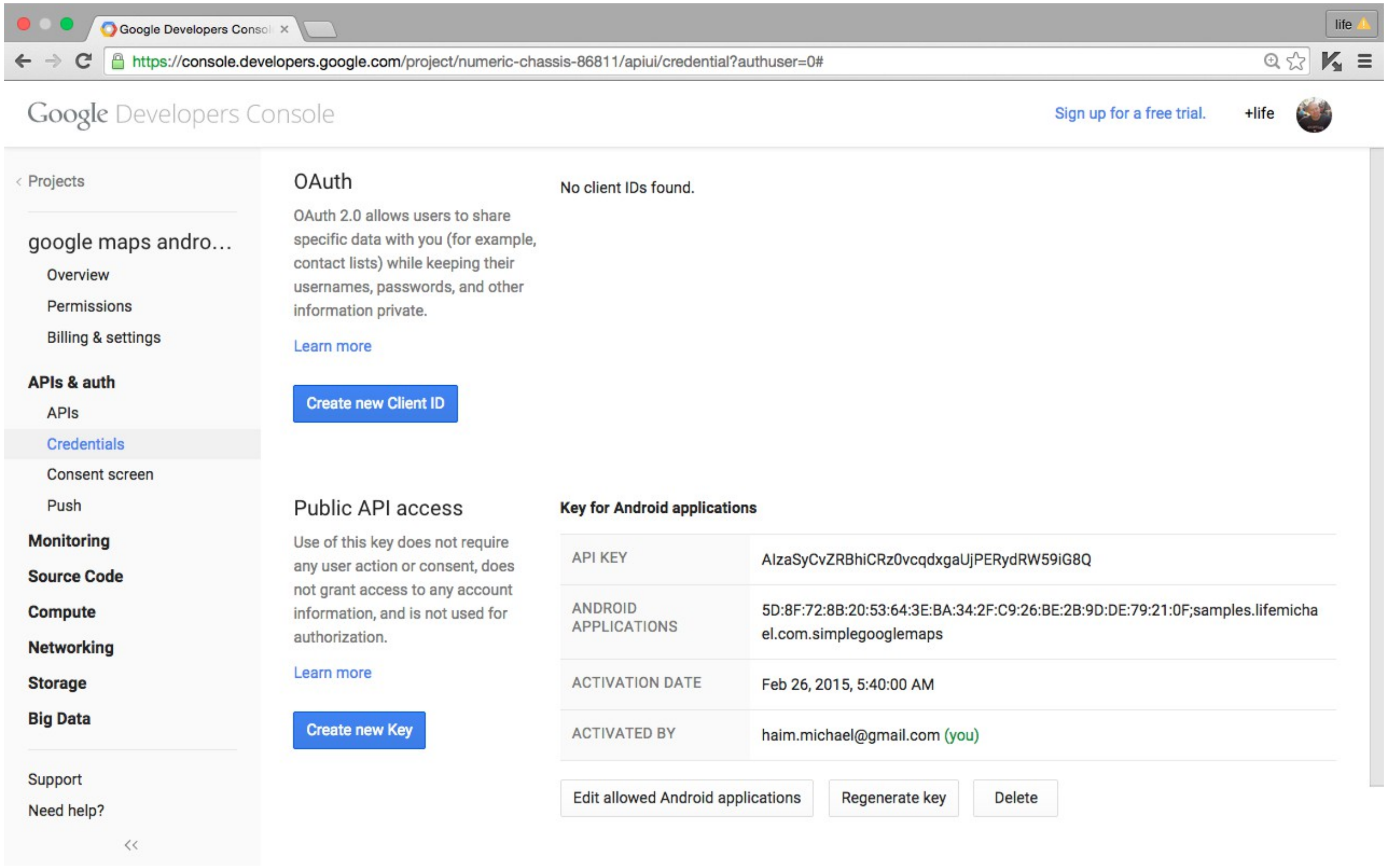
PROJECT NAME	PROJECT ID	REQUESTS	ERRORS	CHARGES
abelski-first-sample	abelski-first-sample	0	0	–
API Project Glass	api-project-919054745835	0	0	–
apo lala	apo-501	6	0	–
gcm_demo	api-project-607307744042	0	0	–
hitandroidmap	api-project-789350286412	0	0	–
justmap	our-lacing-501	0	0	–
life michael google maps demo	api-project-538355485439	0	0	–
myphp	model-choir-307	0	0	–

https://code.google.com/apis/console/

Google Maps API Key

- In order to get the key we just need to create a new project using the details we have, select the Google APIs we want to be available for the new created project and get the credentials (key).
- For using the Google Maps API v2 we should get a Public API access key (of the 'Android Key' type).

Google Maps API Key



OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

Learn more

Create new Client ID

No client IDs found.

Public API access

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

Learn more

Create new Key

Key for Android applications

API KEY	AIzaSyCvZRBhiCRz0vcqdxgaUjPERydRW59iG8Q
ANDROID APPLICATIONS	5D:8F:72:8B:20:53:64:3E:BA:34:2F:C9:26:BE:2B:9D:DE:79:21:0F;samples.lifemichael.com.simplegooglemaps
ACTIVATION DATE	Feb 26, 2015, 5:40:00 AM
ACTIVATED BY	haim.michael@gmail.com (you)

Edit allowed Android applications | Regenerate key | Delete

Google Maps API Key

- In order to update our application with the key we got we just need to add the following meta-data element as a child to the application element in the AndroidManifest.xml file.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="______________"/>
```

Uses Permissions

- We should update the AndroidManifest.xml file adding the following uses permissions:

```
<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE "/>

<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Uses Permissions

- In addition, most likely that we will also need to add the following uses permissions:

```
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION"/>

<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

OpenGL ES Version

- Google Maps Android API v2 uses the OpenGL ES version 2 for rendering the graphics. We should specify this requirement using the <uses-feature> element that should be added as a child to <manifest> element (in the AndroidManifest.xml file).

```
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

Code Sample

- We can now develop a simple demo for showing a map using this API.

Code Sample

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>
```

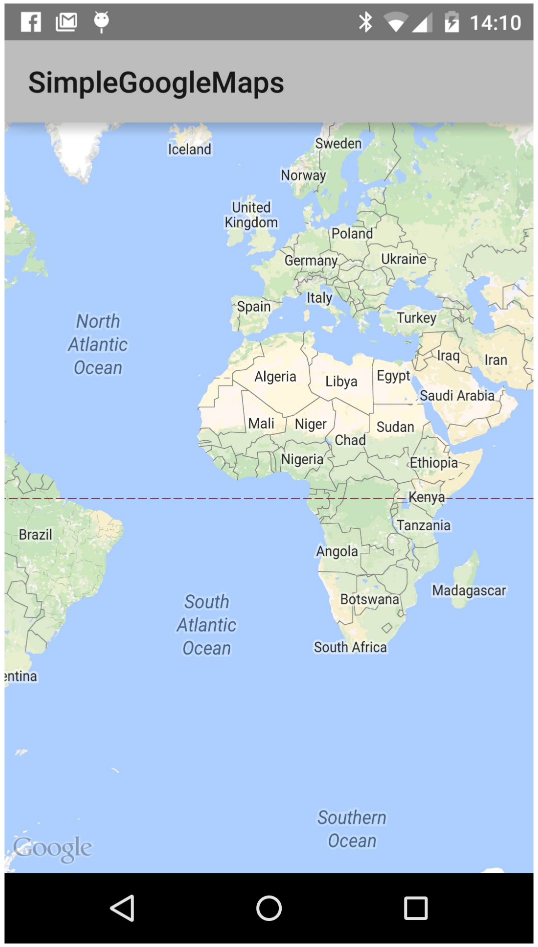
activity_main.xml

Code Sample

```
package samples.lifemichael.com.simplegooglemaps;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

MainActivity.java

Code Sample


14:10
SimpleGoogleMaps
Iceland
Sweden
Norway
United Kingdom
Poland
Germany
Ukraine
Italy
Spain
Turkey
North Atlantic Ocean
Iraq
Iran
Algeria
Libya
Egypt
Saudi Arabia
Mali
Niger
Sudan
Chad
Nigeria
Ethiopia
Kenya
Tanzania
Brazil
Angola
Madagascar
Botswana
South Atlantic Ocean
South Africa
ntina
Southern Ocean
Google