

# Android Location and Maps

Jussi Pohjolainen

# Location Services and Maps

- Android provides location framework that your app can use to determine
  - Device's location
  - Listening for updates
- **Google maps** *external library* available for displaying and managing maps

# **LOCATION SERVICES**

# Location Services

- By using LocationManager you are able to
  - Query for the list of LocationProviders for the last known location
  - Register for updates of the user's current location
  - Register for given Intent to be fired if the device comes within given proximity of given lat/long

# Obtaining User Location

- Location can be determined via GPS and/or cell tower + Wi-Fi signals
- GPS is accurate, but needs outdoors, fix is slower and it uses more battery.
- To request location, you use `LocationManager` - class

# Requesting Location Updates

```
class Something implements LocationListener {  
  
    public void initializeLocation() {  
  
        // Acquire a reference to the system Location Manager  
        LocationManager locationManager =  
            (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);  
  
        // Register the listener with the Location Manager to receive location updates  
        locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,  
                                                0,  
                                                0,  
                                                this);  
  
    }  
  
    public void onLocationChanged(Location location) {  
        // Called when a new location is found by the network location provider.  
        doSomething(location);  
    }  
  
    public void onStatusChanged(String provider, int status, Bundle extras) {}  
  
    public void onProviderEnabled(String provider) {}  
  
    public void onProviderDisabled(String provider) {}  
  
}  
}
```

Can be also  
GPS\_PROVIDER

Control  
the  
frequency  
which  
listener  
receives  
updates;  
min time  
and min  
distance

# Requesting User Permissions

- You must add permissions in order to get user location

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

- ACCESS\_COARSE\_LOCATION
  - If you use NETWORK\_PROVIDER
- ACCESS\_FINE\_LOCATION
  - If you use GPS\_PROVIDER or NETWORK\_PROVIDER

# Getting Last Known Location

- First location can take time. Use cached location!
  - Location lastKnownLocation =  
locationManager.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);



# Mock Location Data

- If you don't have Android device, you can use emulator for location services by giving mock data
- Mock data **works only with GPS\_PROVIDER**
- Use
  - Emulator Control View from Eclipse
  - DDMS (Dalvik Debug Monitor Server)
  - Geo command from console

# Emulator Control View

The screenshot shows the 'Emulator Control' tab in an IDE. It contains three main sections:

- Telephony Status:** Includes dropdown menus for 'Voice' (set to 'home'), 'Speed' (set to 'Full'), 'Data' (set to 'home'), and 'Latency' (set to 'None').
- Telephony Actions:** Features an 'Incoming number' text field, radio buttons for 'Voice' (selected) and 'SMS', a 'Message' text area, and 'Call' and 'Hang Up' buttons.
- Location Controls:** Includes tabs for 'Manual' (selected), 'GPX', and 'KML'. Below these are radio buttons for 'Decimal' (selected) and 'Sexagesimal', followed by input fields for 'Longitude' (-122.084095) and 'Latitude' (37.422006), and a 'Send' button.

# Geo

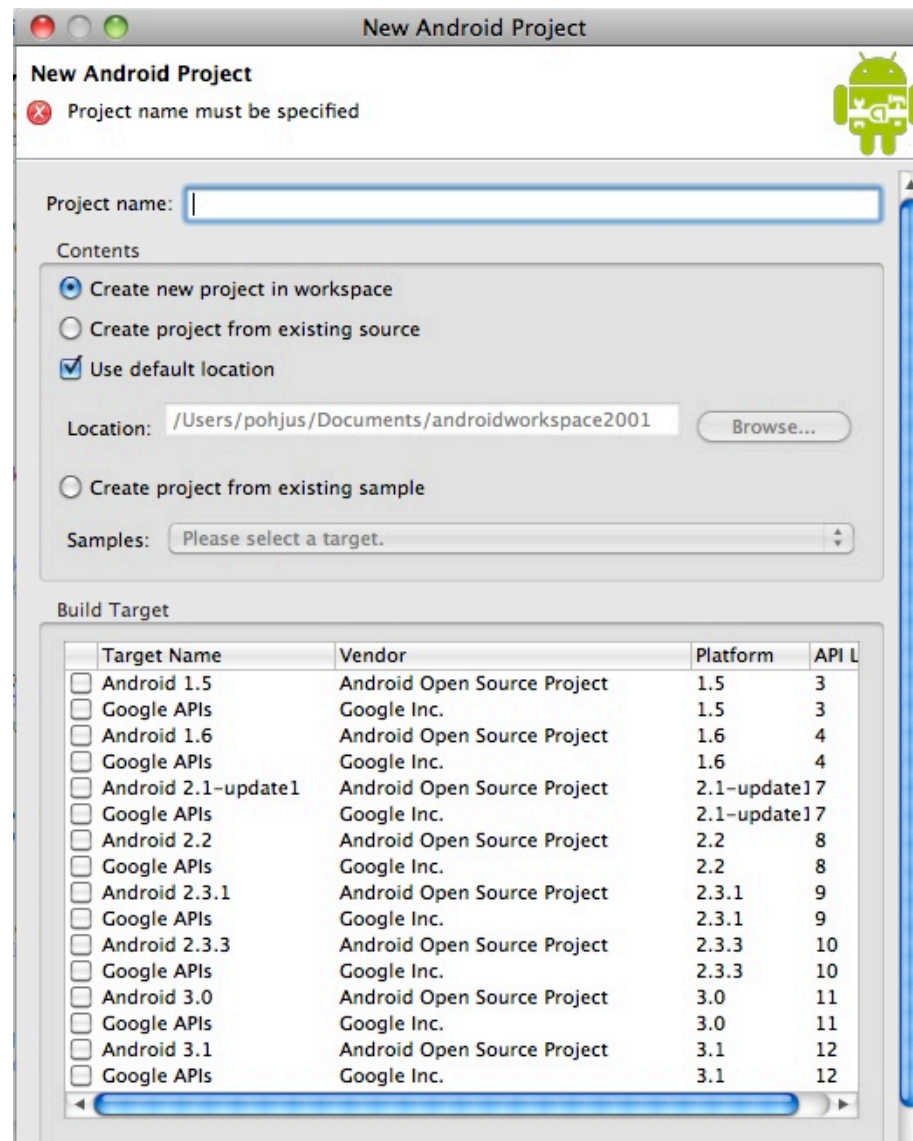
- Connect to emulator from console
  - `telnet localhost <console-port>`
- Send the location data
  - `geo fix -121 46`

# **GOOGLE MAPS EXTERNAL DIRECTORY**

# Google Maps

- **External API Add-On** to Android SDK
- Install Google APIs Add-On from Android SDK and AVD Manager (Google APIs by Google)
- When developing, set Google API Add-On as target

# Google APIs as Target



# Overview

1. Add uses-library and internet permission to manifest file
2. Use the Maps API
3. Get Maps API key and sign your app

# 1. Add uses-library element to Manifest file

- Because we're using the Google Maps library, which is not a part of the standard Android library, we need to declare it in the Android Manifest
  - `<uses-library  
    android:name="com.google.android.maps" />`
- Internet permissions (downloadable maps)
  - `<uses-permission  
    android:name="android.permission.INTERNET  
    " />`



## 2. Use the Maps API: MapView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainlayout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="Your Maps API Key"
    />

</RelativeLayout>
```

## 2. Use the Maps API: Class

```
public class MyMapView extends MapActivity {  
  
    MapView mapView;  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.mymapview);  
  
        mapView = (MapView) findViewById(R.id.mapview);  
  
        // Add zoom functionality  
        mapView.setBuiltInZoomControls(true);  
    }  
  
    @Override  
    protected boolean isRouteDisplayed() {  
        // TODO Auto-generated method stub  
        return false;  
    }  
}
```

---

### 3. Get Maps API key and Sign Your App

- Create MD5 certificate fingerprint either in release or in debug
- Release
  - `$ keytool -list -alias alias_name -keystore my-release-key.keystore`
- Debug
  - `$ keytool -list -alias androiddebugkey -keystore <path_to_debug_keystore>.keystore -storepass android -keypass android`
- Path to debug keystore in Windows Vista
  - `C:\Users\<user>\.android\debug.keystore`

# 3. Sign with the Service

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

## Android Maps API Key Signup

### Sign Up for the Android Maps API

The Android Maps API lets you embed [Google Maps](#) in your own Android applications. A single Maps API key is used for all applications. See the [documentation page](#) for more information about application signing. To get a Maps API key for your certificate obtained using Keytool. For example, on Linux or Mac OSX, you would examine your debug keystore like this:

```
$ 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 certificate 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

#### 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 document as the "Maps API(s)" or the "Service") is subject to the terms of a legal agreement between you and

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

My certificate's MD5 fingerprint:

[Generate API Key](#)

# Showing Latitude and Longitude on Map

```
mapView = (MapView) findViewById(R.id.mapview);  
mapController = mapView.getController();
```

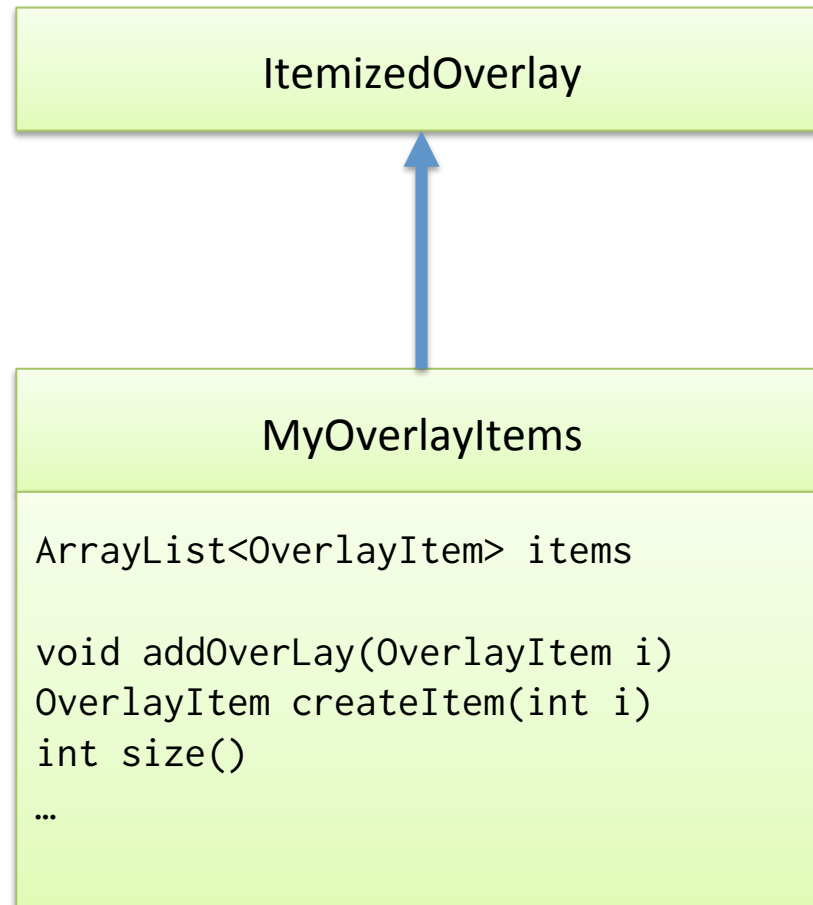
```
int lat = (int) (location.getLatitude() * 1E6);  
int lng = (int) (location.getLongitude() * 1E6);
```

```
GeoPoint point = new GeoPoint(lat, lng);  
mapController.animateTo(point);
```

# **DISPLAYING GRAPHICS ON MAPS**

# Custom Markers

- Overlay – *Individual item on map*
- Create custom class that inherits ItemizedOverlay class
- ItemizedOverlay class is a base class for an Overlay which consists of a list of OverlayItems.





```

public class MyMarkers extends ItemizedOverlay {

    private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();

    public MyMarkers(Drawable defaultMarker) {

        // The constructor must define the default marker for each of the
        // OverlayItems.
        // In order for the Drawable to actually get drawn, it must have its
        // bounds defined. Most commonly, you want the center-point at the
        // bottom of the image to be the point at which it's attached to the map
        // coordinates. This is handled for you with the boundCenterBottom()
        // method.
        super(boundCenterBottom(defaultMarker));
    }

    // In order to add new OverlayItems to the ArrayList
    public void addOverlay(OverlayItem overlay) {
        mOverlays.add(overlay);
        populate();
    }

    @Override
    protected OverlayItem createItem(int item) {
        return mOverlays.get(item);
    }

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

```

# From Activity

```
// All overlay elements on a map are held by the MapView, so when you want to
// add some, you have to get a list from the getOverlays() method.
List<Overlay> mapOverlays = mapView.getOverlays();

// Create drawable (the image to be shown on the map)
Drawable drawable = this.getResources().getDrawable(R.drawable.androidmarker);

// Create instance of your class and pass the default drawable
MyMarkers mymarkers = new MyMarkers(drawable);

// Create geopoint
GeoPoint point = new GeoPoint(1924300,-99120444);

// Create item to that point
OverlayItem overlayitem = new OverlayItem(point, "title", "text");

// Add item to collection
mymarkers.addOverlay(overlayitem)

// Add the collection to map
mapOverlays.add(itemizedoverlay);
```

