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(Contex     ATION_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

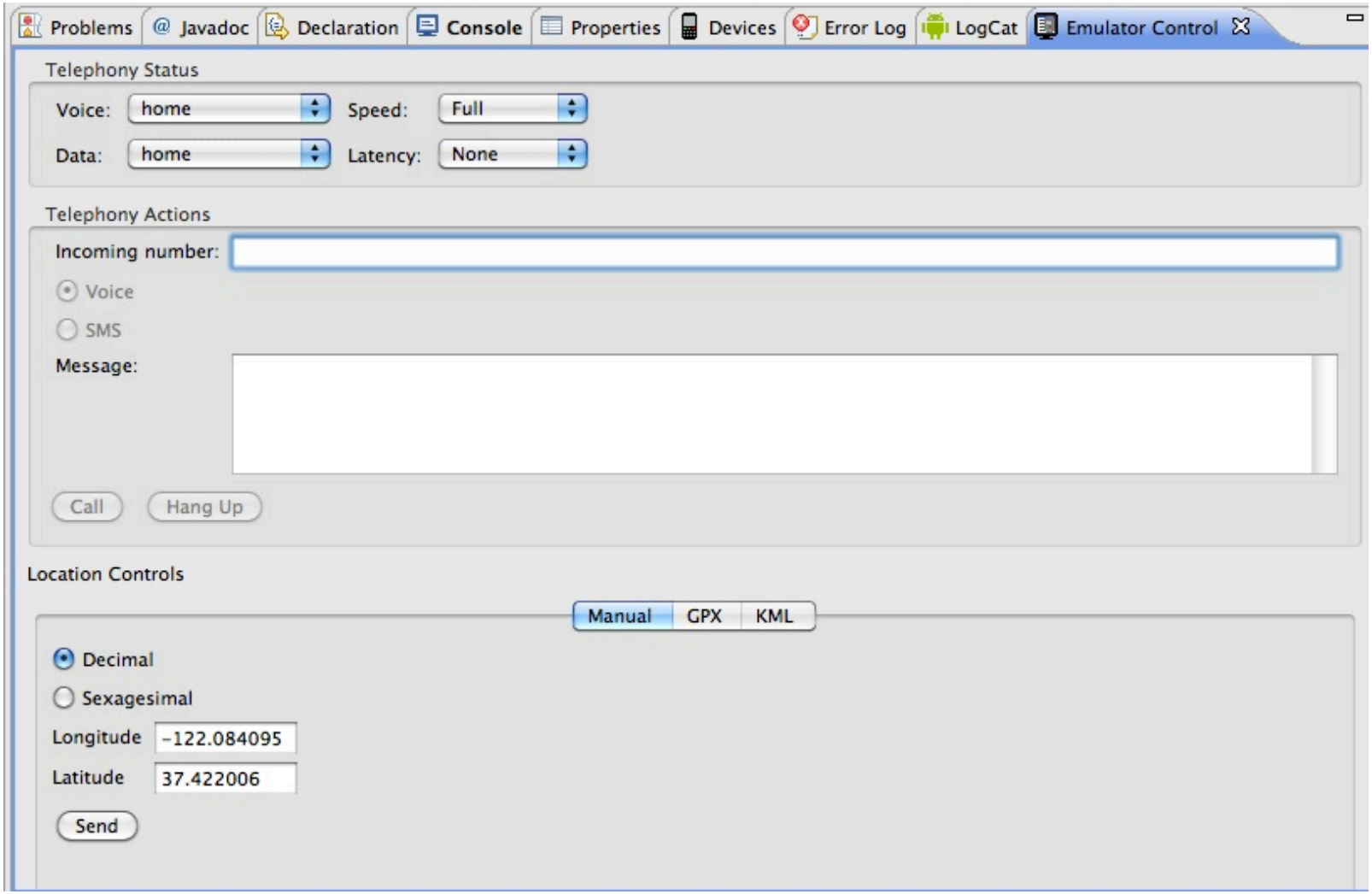
Getting Last Known Location

- 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



Geo

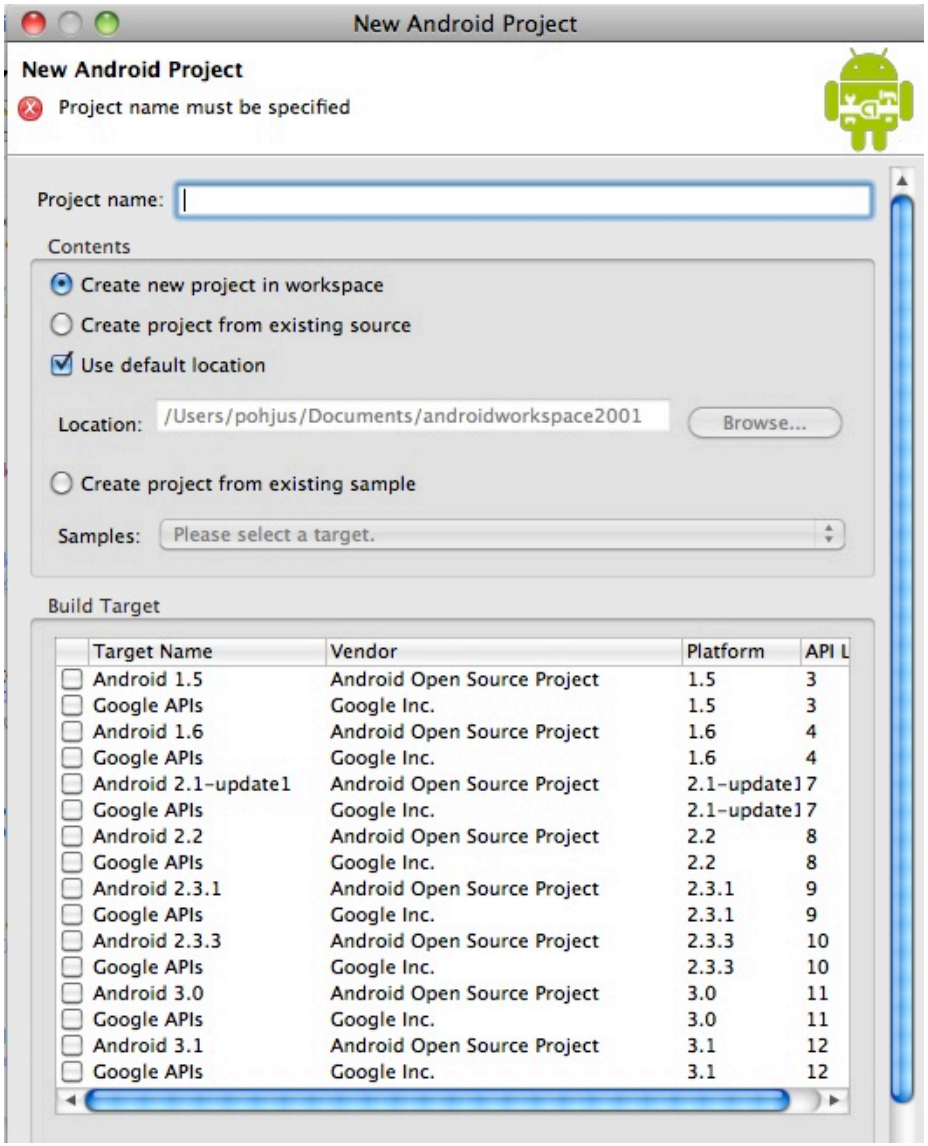
- Connect to emulator from console
  - `telnet localhost <console-port>`
- Send the location data
  - `geo fix -121 46`

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

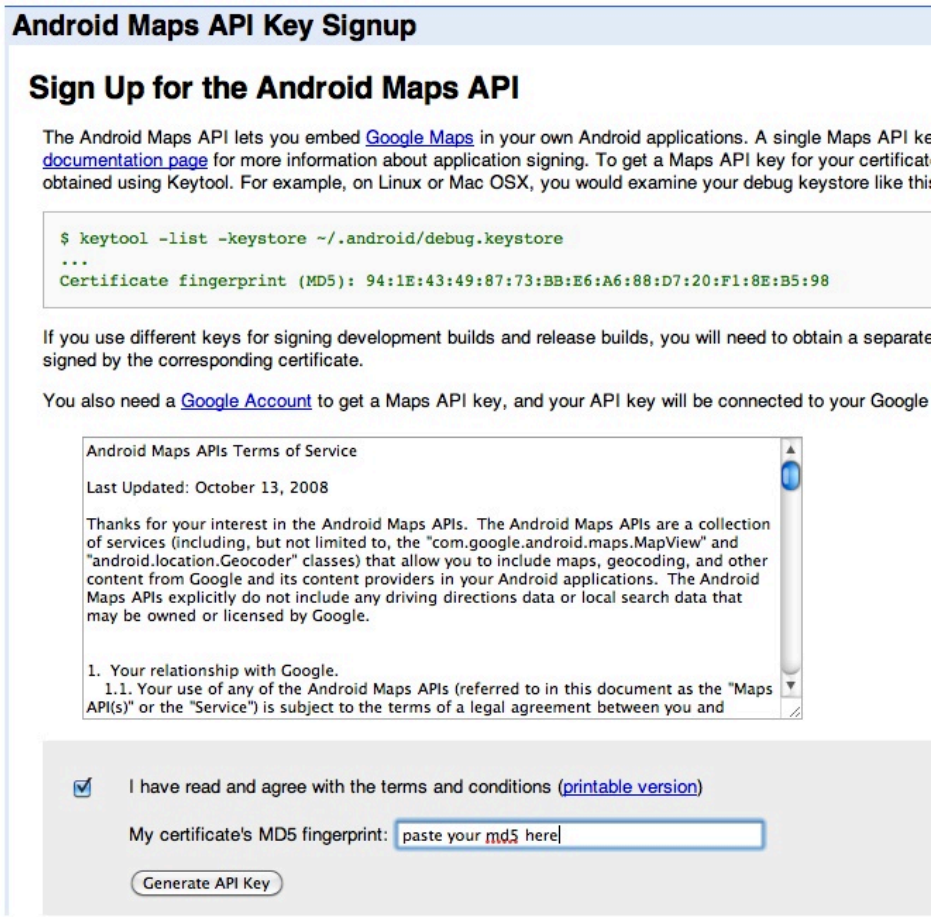
```
public class MyMapView extends MapActivity {

    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



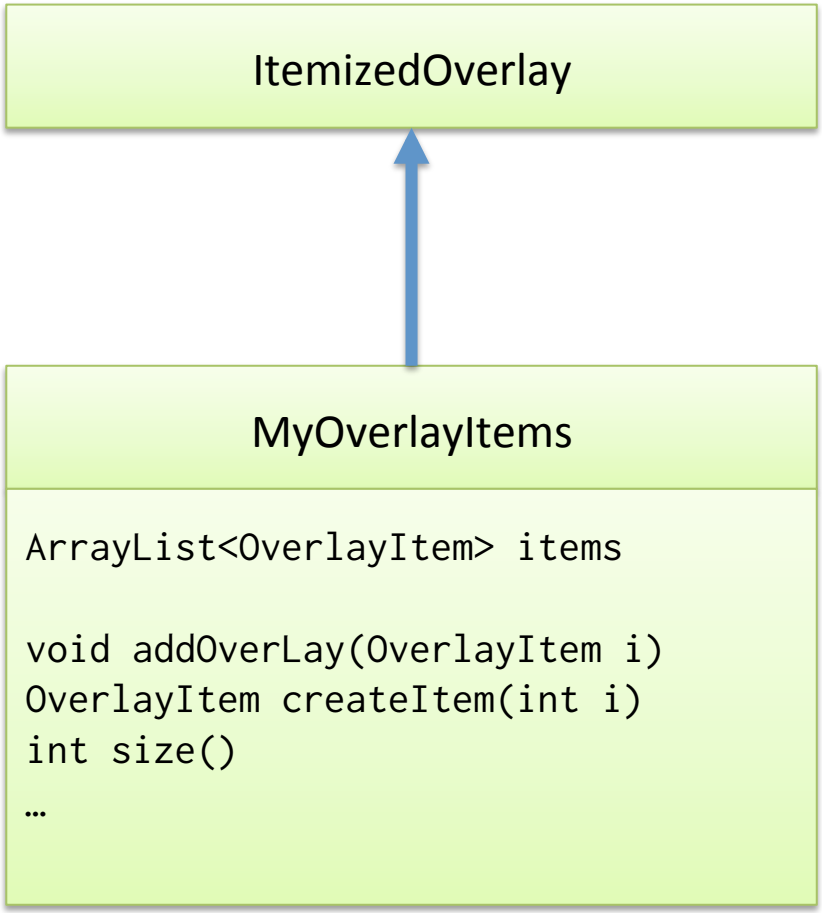
Showing Latitude and Longitude on Map

```
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 inherites `ItemizedOverlay` class
- `ItemizedOverlay` class is a base class for an Overlay which consists of a list of OverlayItems.


ItemizedOverlay
MyOverlayItems
ArrayList<OverlayItem> items
void addOverLay(OverlayItem i)
OverlayItem createItem(int i)
int size()
…

```
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);
```


Oregon
Idaho
Salt Lake City
Nevada
Utah
San Francisco
Stockton
California
Bakersfield
Las Vegas
Los Angeles
Arizona
Riverside
Phoenix