Location, Maps, Servizi di Sistema

Introduzione

In questa lezione ci occuperemo di analizzare due casi d'uso che fanno uso dei servizi di localizzazione e della Google Maps

LocationManager Project

Nel primo caso d'uso svilupperemo un progetto che ci permette di analizzare le potenzialità del LocationManager di Android

LocationManager 1/2

- LocationManager è una classe che permette di accedere al system location service consentendo di ricevere notifiche sulla posizione del disposivo
- Permette ad una applicazione di ricevere aggiornamenti periodici della posizione del device
- Può invocare un Intent nel caso in cui il dispositivo si trova in una particolare posizione
- Per accedere a questo servizio si usa:

```
public abstract
getSystemService(String name)
```

In cui name assume il valore Context.LOCATION_SERVICE

LocationManager 2/2

- Il LocationManager ottiene le informazioni della posizione mediante i LocationProvider disponibili all'interno del sistema: al momento esistono provider basati su GPS o sulla rete WIFI
- La scelta del provider dipende dai particolari criteri richiesti dalla nostra applicazione, per es. accuratezza, potenza, velocità, costo, altitudine.
- Per essere notificati sulla posizione è necessario definire una istanza dell'interfaccia LocationListener
- Location Manager e LocationListener sono definiti all'interno del package android.location

Location Listener

- Per ricevere una notifica da parte del provider prescelto è necessario definire il seguente metodo:

```
public void
requestLocationUpdates(String provider,
long minTime, float minDistance,
LocationListener listener)
```

- LocationListener rappresenta una interfaccia che prevede la definizione di una serie di operazioni, tra cui:

- onLocationChanged()
- onStatusChanged()
- onProviderEnabled()
- onProviderDisabled()

LocationManager Codice 1/2

```
LocationManager locationManager =
(LocationManager)
getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(LocationMa
nager.GPS_PROVIDER, 0,0,listenerFine);
```

- Mediante questo codice il LocationManager consente di registrare il LocationListener listenerFine rispetto al GPS_PROVIDER
- Avremmo potuto registrarlo anche sul NETWORK_PROVIDER
- Avremmo potuto anche invocare la più generica

```
LocationManager.requestLocationUpdates(locationManage
r.getBestProvider(fineCriteria, true),0, 0,
listenerFine);
```

LocationManager Codice 2/2

```
new LocationListener() {
@Override
public void onLocationChanged(Location location)
{}
@Override
public void onProviderDisabled(String provider)
{}
@Override
public void onProviderEnabled(String provider) {}
@Override
public void onStatusChanged(String provider, int
status,Bundle extras) {} }
```

AndroidManifest.xml

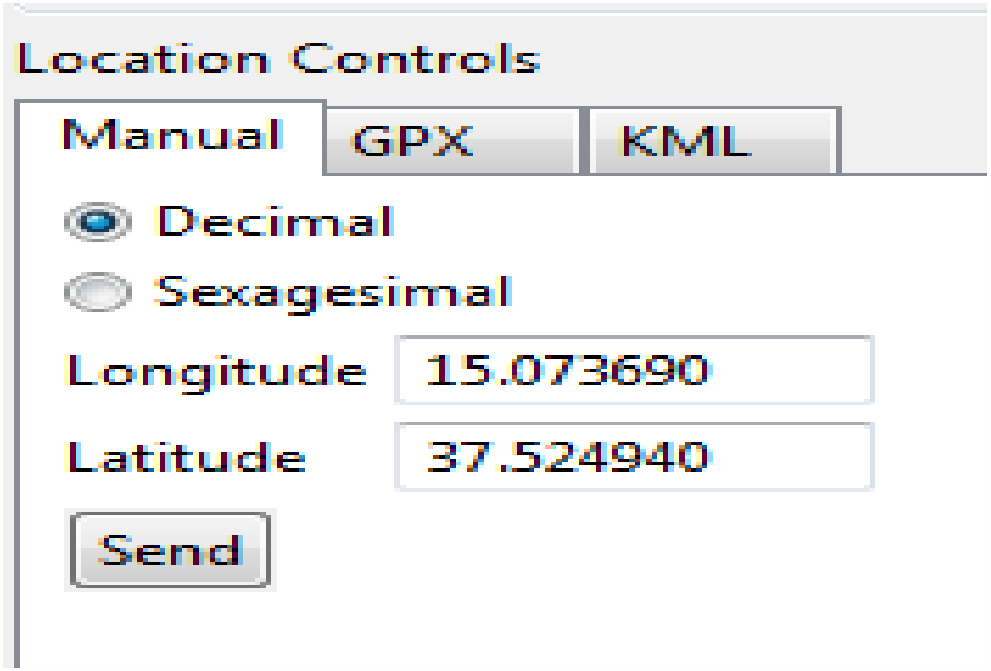
- Per poter usufruire dei servizi di notifica è necesserio richiedere il ”permesso” mediante il seguente codice:

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

- In questo caso stiamo richiedendo di accedere alle informazioni di localizzazione in maniera precisa (FINE), utilizzando quindi il GPS
- In alternativa avremmo potuto richiedere il permesso ACCESS_COARSE_LOCATION per accedere alle informazioni di localizzazione (meno precise) fornite dalla rete WIFI

LocationManager e ADT

- Una caratteristica di questa applicazione è rappresentata dalla possibilità di interagire con il dispositivo virtuale mediante l' Emulator Control



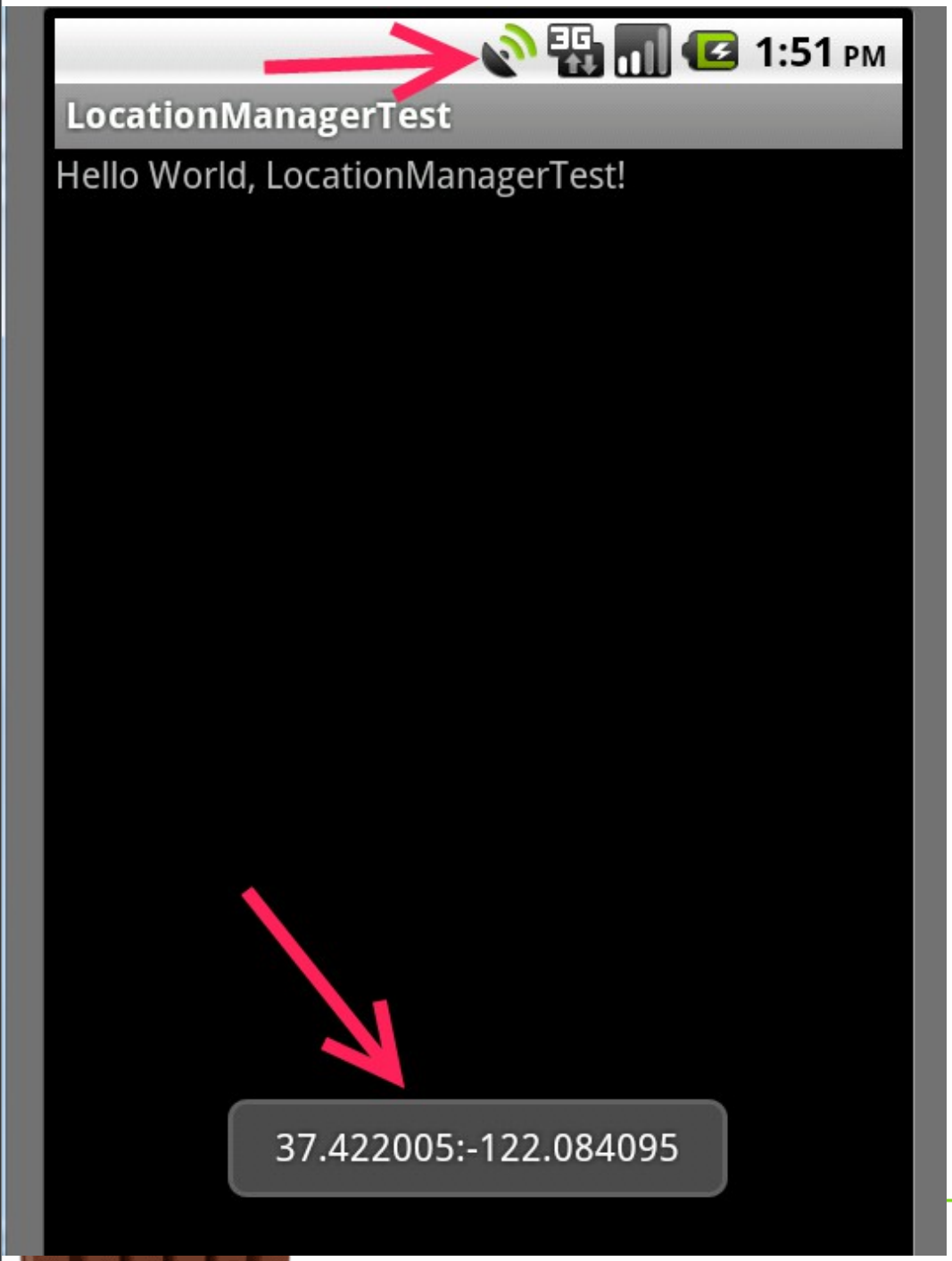
- E' possibile inviare informazioni all'AVD anche nel formato GPX (GPS Exchange Format) e KML (Keyhole Markup Language)

OnLocationChanged 1/3

```
@Override
public void onLocationChanged(Location location)
{
   Log.i(LOG_TAG,location.getLatitude()
+":"+location.getLongitude());
Toast.makeText(LocationManagerTest.this,location.
getLatitude()+":"+location.getLongitude(),
Toast.LENGTH_SHORT).show();
}
```

- In questo caso onLocationChanged registra i valori relativi alla nuova Location mediante un Log e la visualizzazione di un Toast

OnLocationChanged 2/3



onLocationChanged 3/3

- Una seconda particolarità di questa applicazione è la costituita dal fatto che anche quando la nostra Activity non si trova nel ForeGround rimane comunque in ascolto della possibile variazione di posizione
- Ciò significa che l'Activity riceve comunque le notifiche da parte del LocationManager pur essendo in background
- Se questo comportamento non è richiesto da parte della nostra applicazione è conveniente intervenire evitando in tal modo il consumo della batteria

removeUpdates

- Per interrompere le notifiche dovremo quindi richiamare removeUpdates in corrispondenza del metodo onPause()

```
@Override
protected void onPause() {
locationManager.removeUpdates(locationListener
);
super.onPause();
}
```

- Per consentire alla nostra applicazione di ricevere nuovamente le notifiche non appena tornerà in ForeGround dovremo invocare locationManager.requestLocationUpdates all'interndo del metodo onResume()

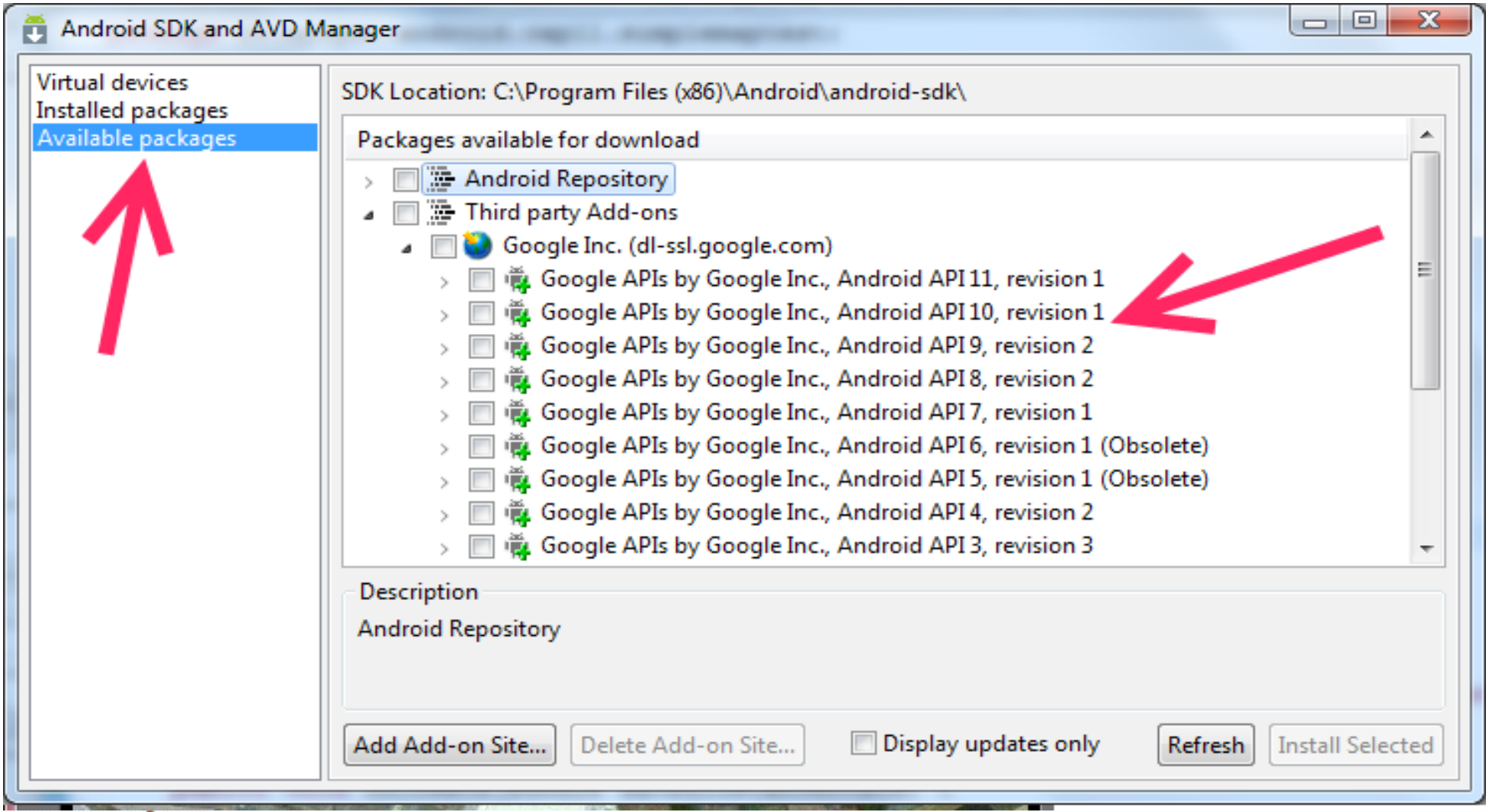
Google Maps API Project

Nel second caso d'uso svilupperemo un progetto basato sulle Google Maps API mediante il quale effettueremo alcuni test sulla geolocalizzazione in Android

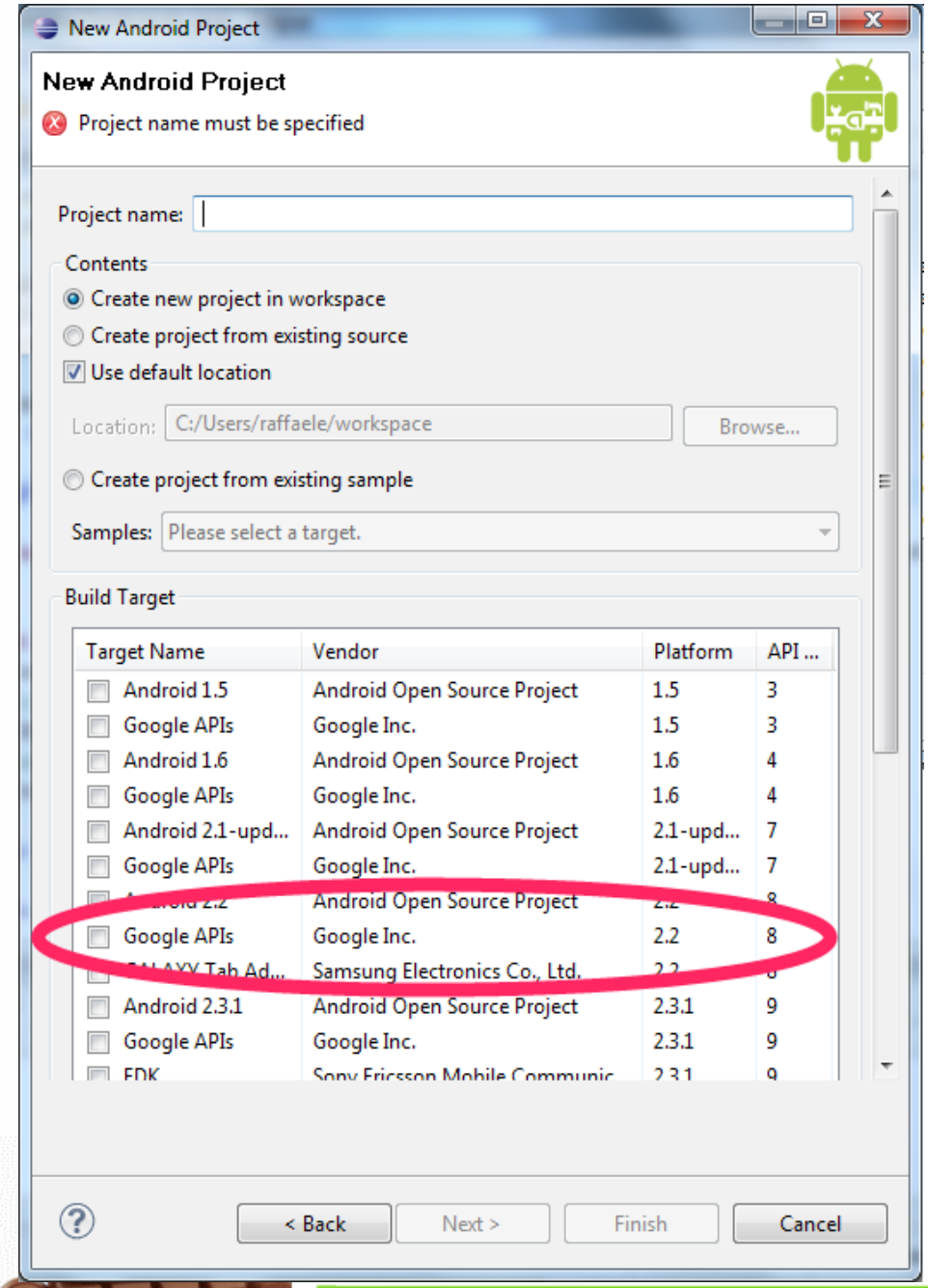
Google Map API 1/3

- Per sviluppare un progetto Android che interagisce con le Map di Google è necessario utilizzare le Google Maps API
- E' necessario creare un AVD ed un progetto che tengano conto del particolare target di riferimento (stiamo utilizzando librerie esterne ad Android standard
- Verifichiamo la disponibilità di queste librerie lanciando il tool ”Android SDK and AVD Manager” ed eventualmente provvediamo alla loro installazione

Google Maps API 2/3



Google Maps API 3/3



- Dopo aver installato le librerie necessarie i nuovi target saranno finalmente disponibili in fase di creazione di un nuovo progetto Android

Google Maps Android v2

- A partire dal 2013 è cambiata la gestione delle Map Key in Android (dalla v1 alla v2):

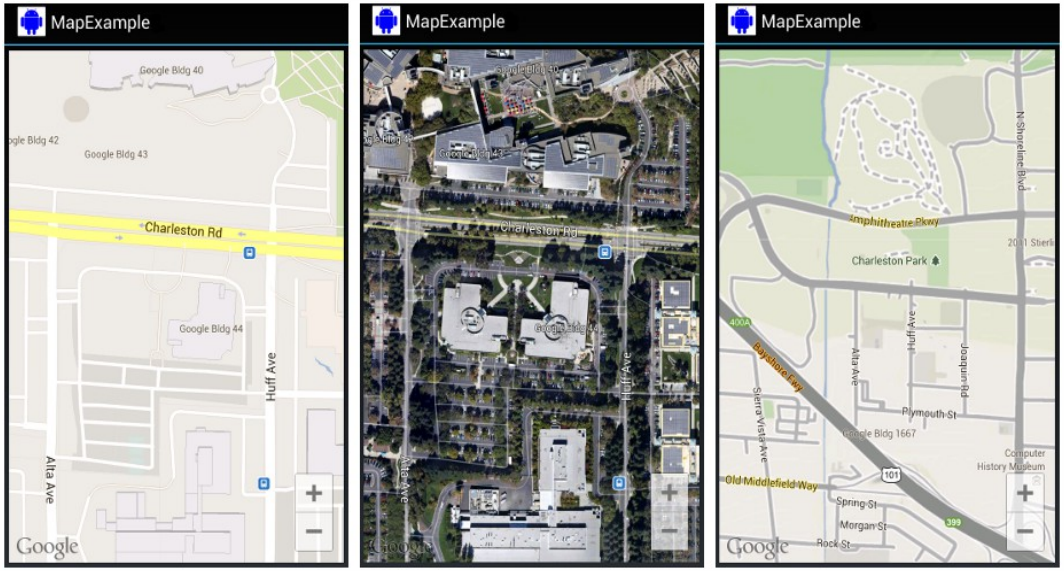
  https://developers.google.com/maps/documentation/android/

- Una guida utile per configurare l'emulatore è:

  http://hmkcode.com/run-google-map-v2-on-android-emulator/

- Per partire...

  https://developers.google.com/maps/documentation/android/map

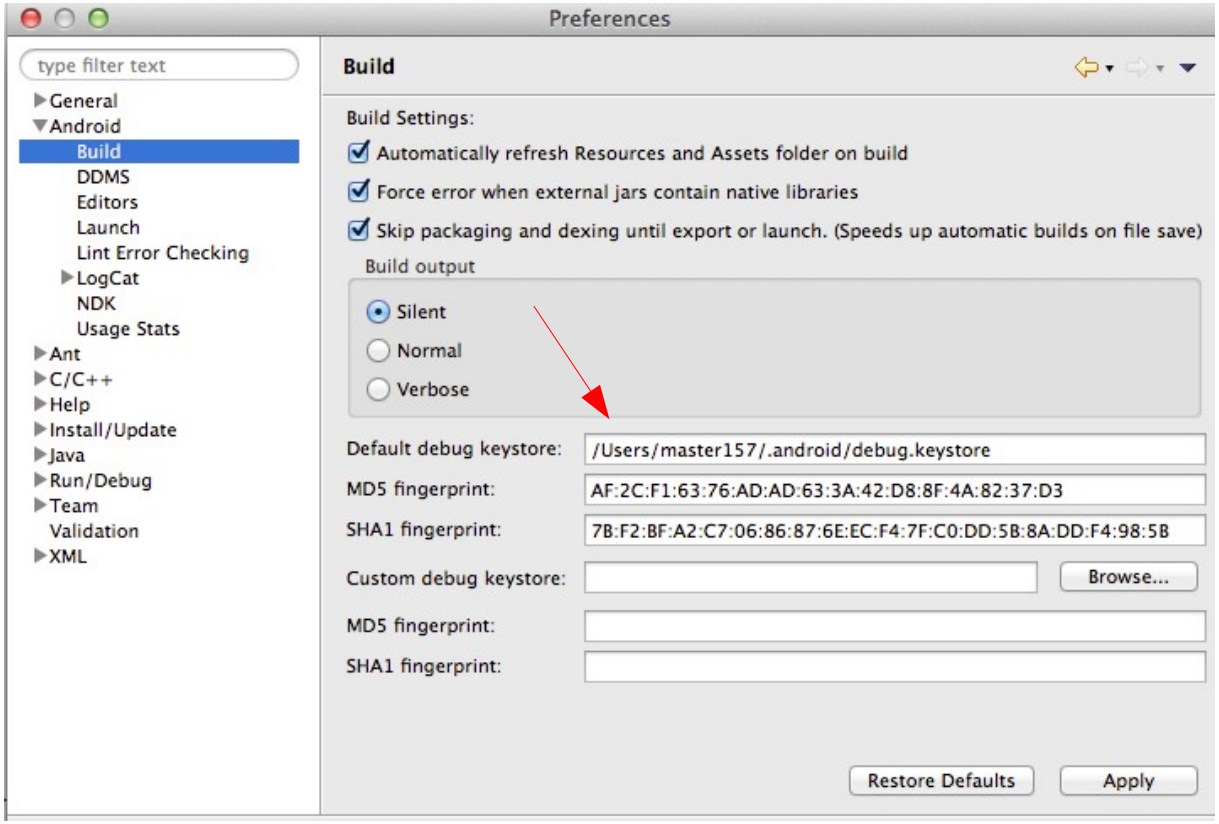


map key 1/4

- Per poter utilizzare le Google Maps API è necessario disporre di una map-key
- Tale codice sarà associato al certificato che utilizziamo per firmare la nostra applicazione
- E' necessario disporre di una chiave per il certificato di debug ed una per il certificato di release
- Di seguito vedremo come ottenere il codice per il certificato di debug

map key 2/4

- Il primo passo consiste nell'individuare dove si trova il keystore
- Per far questo, in Eclipse, andiamo su Window| Preferences|Build
- Oppure in ADT/Preferenze



map key 3/4

- Facendo riferimento al keystore eseguiamo il seguente comando:

```
keytool -list -alias androiddebugkey -keystore <keystore>
-storepass android -keypass android
```

- Il risultato di questo comando sarà la visualizzazione dell'MD5

```
Tipo keystore: JKS
Provider keystore: SUN
Il keystore contiene 1 entry
androiddebugkey, 11-mar-2011, PrivateKeyEntry,
Impronta digitale certificato (MD5):
94:1E:43:49:87:73:BB:E6:A6:88:D7:20:F1:8E:B5:98
```

map key 4/4

- Accedendo alla Google APIs Console:

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

- Abilitare il servizio Google Map Android API v2
- Andare su API Access>>Create New Android Key
- Copiare l'SHA-1 fingerprint e il package.name separati da ";"

  BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;com.example.android.mapexample

- Al termine viene fornita la Key for Android apps da utilizzare nella nostra App:

  AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0

AndroidManifest.xml

```
 <permission
   android:name="com.example.demo.map.permission.MAPS_RECEIVE"
   android:protectionLevel="signature" />

 <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-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
/>
<!-- The following two permissions are not required to use
     Google Maps Android API v2, but are recommended. -->
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Google Maps demo

```
//Focalizziamo la mappa su un punto prefissato
googleMap.moveCamera(CameraUpdateFactory.newLatLn
gZoom(DIEEI, 15));

//Aggiungiamo un marker
Marker dieei = googleMap.addMarker(new
MarkerOptions().position(DIEEI).title("DIEEI"));
```

- La mappa è presente come fragment nel layout
- Quando non è definita alcuna posizione iniziale l'applicazione visualizzerà la mappa di default

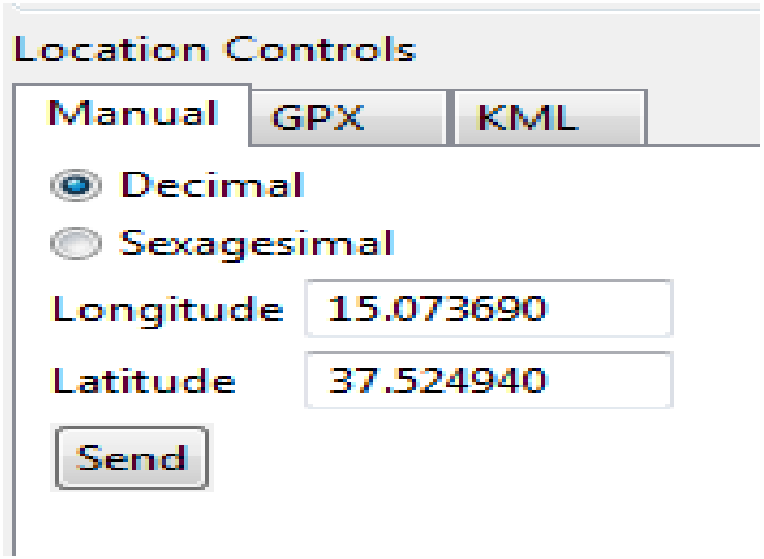
Google Maps demo

```
new LocationListener() {
@Override
public void onLocationChanged(Location location) {
}
@Override
public void onProviderDisabled(String provider) {
}
@Override
public void onProviderEnabled(String provider) {
}
@Override
public void onStatusChanged(String provider, int status,Bundle extras) {
}
});
```

- Come nel caso precedente definiamo i metodi del LocationListener che intendiamo gestire

Google Maps den

- Utilizziamo i seguenti valori ed inviamoli all'AVD



- Saranno visualizzati sulla mappa

I servizi di Sistema

- Negli esempi proposti in precedenza si è fatto più volte uso dei servizi di sistema.
- Abbiamo sempre effettuato l'accesso con la chiamata:

  getSystemService()

  che ci ha consentito di utilizzare uno specifico Manager
- Oltre a quelli già analizzati, come il NotificationManager, l'AlarmManager

I servizi di Sistema

- Power Service: per la gestione del risparmio energetico. Si accede mediante POWER_SERVICE
- Key Guard Service: per la gestione del lock della tastiera. Si accede mediante KEYGUARD_SERVICE
- Vibrator Service: per la gestione della vibrazione del dispositivo. Si accede mediante VIBRATOR_SERVICE
- Alarm Service: per la gestione degli allarmi. Si accede mediante ALARM_SERVICE

I servizi di Sistema

- Audio Service: per la gestione delle risorse audio. Si accede mediante AUDIO_SERVICE
- Telephony Service: per la gestione di chiamate ed SMS. Si accede mediante TELEPHONY_SERVICE
- Connectivity Service: per la gestione della rete. Si accede mediante CONNECTIVITY_SERVICE
- WiFi Service: per la gestione della WIFI. Si accede mediante WIFI_SERVICE

I servizi di Sistema

- Accessibility Service: per la gestione eventi GUI. Si accede mediante ACCESSIBILITY_SERVICE
- Input Method Service: per la gestione dell'interattività. Si accede mediante INPUT_METHOD_SERVICE
- ClipBoard Service: per la gestione degli appunti. Si accede mediante CLIPBOARD_SERVICE

I Sensor Service 1/2

- Rappresentano probabilmente i servizi più specifici di Android
- Si accede mediante SENSOR_SERVICE
- E' possibile accedere ad una lunga lista di sensori, tra cui l'accelerometro, il giroscopio e la bussola.

I Sensor Service 1/2

TYPE_ACCELEROMETER Accelerometro.

TYPE_ALL Tutti.

TYPE_AMBIENT_TEMPERATURE Temperatura

TYPE_GRAVITY Gravità.

TYPE_GYROSCOPE Giroscopio

TYPE_LIGHT Luminosità

TYPE_LINEAR_ACCELERATION Accelerazione lineare.

TYPE_MAGNETIC_FIELD Intensità campo magnetico

TYPE_PRESSURE Pressione

TYPE_PROXIMITY Prossimità

I Sensor Service 1/2

TYPE_RELATIVE_HUMIDITY Umidità

TYPE_ROTATION_VECTOR Rotazione vettoriale

TYPE_ORIENTATION (Bussola) Deprecato

Utilizzo dei Sensor Service

```
@Override
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_FASTEST);
    mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
SensorManager.SENSOR_DELAY_FASTEST);
    mSensorManager.registerListener(this,
 mSensorManager.getDefaultSensor(Sensor.TYPE_TEMPERATURE),
SensorManager.SENSOR_DELAY_FASTEST);
}

@Override
protected void onStop() {
    mSensorManager.unregisterListener(this);
    super.onStop();
}
```