Workshop – werken met google maps in android

1. Instellingen

1.1 Niew project
Maak een nieuw project aan.

1.2 Manifest instellingen
Om verbinding te maken :

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

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="leonard_en_alwin.ch.MapsWorkshop"
    android:versionCode="1"
    android:versionName="1.0.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".MapsWorkshop"
          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-permission android:name="android.permission.INTERNET" />
</manifest>
```

AndroidManifest.xml

Stap 3 – Een apikey genereren
Hier: http://code.google.com/android/maps-api-signup.html kan je een apikey genereren. Dit wordt gedaan aan de hand van de MD5 Fingerprint van je Certificaat. In de meeste gevallen kan je je certificaat hier terug vinden:

- Windows Vista: C:\Users\<user>\AppData\Local\Android\debug.keystore
- Windows XP: C:\Documents and Settings\<user>\Local Settings\Application Data\Android\debug.keystore
- OS X and Linux: ~/.android/debug.keystore

Je MD5 Fingerprint kan je met de Keytool achterhalen (Linux):

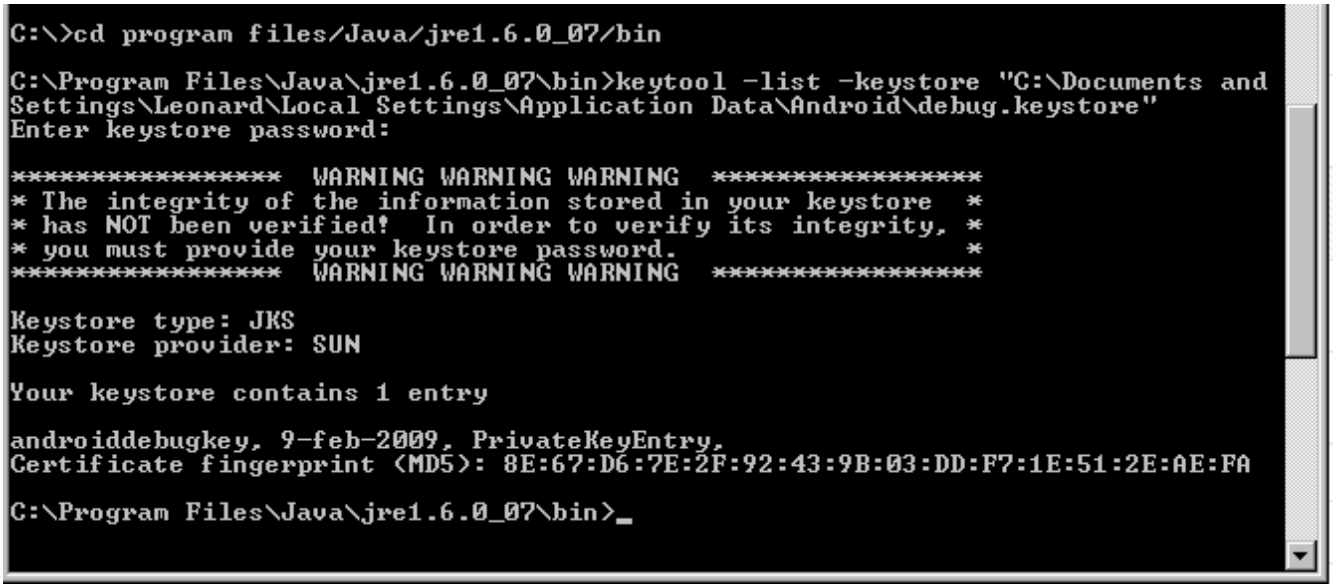
- type dit in je konsole: keytool -list -keystore ~/.android/debug.keystore

Je MD5 Fingerprint kan je met de Keytool achterhalen (Windows):

1. Open cmd.
2. Ga naar de bin map van de java sdk die je gebruikt. Bijvoorbeeld: C:\>cd program files/Java/jre1.6.0_07/bin.
3. Open je keystore met de keytool op de manier: C:\Program Files\Java\jre1.6.0_07\bin>keytool -list -keystore "C:\Documents and Settings\Leonard\Local Settings\Application

Data\Android\debug.keystore"

4. De keytool zal dan om een wachtwoord vragen. Als je nooit specifiek een wachtwoord daarvoor hebt ingevuld kan je gewoon op enter drukken.
5. Kopieer de MD5 fingerprint in het signup formulier voor je google maps apikey.



Stap 4 – layout instellen
Verwijder de standaard genereerde TextView uit de layout en voeg een MapsView toe binnen de LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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="0F0FZc91zp6NeJMbHfSr8dzSj5xeGflk22TKlHg"
   />

</LinearLayout>
```

main.xml

2. MapActivity

2.1 MapActivity extends

Inplaats van het extenden van Activity wat je normaal doet bij het maken van een nieuwe activity moet je bij Activity die een map gebruikt de MapActivity extenden. De MapActivity vereist dat je een extra

methode implementeert, die we overigens verder niet nodig hebben.

Als het goed is ziet je code er nu zo uit als hieronder en moet het al mogelijk zijn een map in beeld te krijgen als je het project build.

```
package leonard_en_alwin.ch.MapsWorkshop;

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

public class MapsWorkshop extends MapActivity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }

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

2.2 Controls

Natuurlijk heb je nog niet zoveel aan een map als je er niks mee kan. Daarom gaan we controls aan de map toevoegen waarmee je kan in- en uitzoomen. De zoom controls moeten worden toegevoegd aan een view. Die view gaan we eerst aanmaken in de main.xml, voeg deze code toe aan de main.xml onder de mapview binnen en binnen de LinearLayout waar ook de mapview in staat:

```
   <LinearLayout
   android:id="@+id/zoomview"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBottom="@id/mapview"
   android:layout_centerHorizontal="true"/>
```

We zeggen hier met alignBottom="@id/mapview" dat de layout zicht onderaan de mapview positioneert en met centerhorizontal zeggen we dat we de controls gecentreerd willen hebben in het midden van het scherm.

Nu kunnen we de controls gaan toevoegen aan onze Activity. Voeg de onderstaande functie toe en roep hem aan in de Oncreate() functie.

```
protected void addZoomControls() {
       LinearLayout controlsView = (LinearLayout) findViewById(R.id.controlsView);
       MapView mapView = (MapView) findViewById(R.id.mapview);
       ZoomControls zoomControls = (ZoomControls) mapView.getZoomControls();

       controlsView.addView(zoomControls);
}
```

3. Markers en Overlays

3.1 Nieuwe class aan maken voor de overlays
Maak een nieuwe class aan die de ItemizedOverlay extend.

3.2 Maak een nieuwe class variable aan.
```
private ArrayList<OverlayItem> overlays = new ArrayList<OverlayItem>();
```

Deze ArrayList gaat alle OverlayItems bevatten die we gaan toevoegen

3.3 Contructor
In de contructor roepen we de constructor in de superclass aan. We geven de type marker mee die we gaan gebruiken.

```
public CustomItemizedOverlay(Drawable defaultMarker) {
      super(boundCenterBottom(defaultMarker));
}
```

```
We geven hier aan hoe de marker aan de map word "verankerd".
```

```
public void addOverlay(OverlayItem overlay) {
      overlays.add(overlay);
    populate();
}
```

3.4 Functie om Items toe te voegen aan de Arraylist
De functie populate die we hier aan roepen zorgt er voor dat alle Items klaar worden gemaakt om op de map te worden getekend.

3.5 Verplichte functies die overschreven moeten worden
```
@Override
protected OverlayItem createItem(int i) {
  return overlays.get(i);
}

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

De functie createItem(int i) zorgt er voor dat de Items op de juiste manier worden uitgelezen uit de ArrayList.

De functie size() returned het aantal items dat in de ArrayList staat.

4. Overlays op de map tekenen

Maak eerst een nieuwe functie aan in de MapView class die het uitvoeren van de Overlays afhandeld.

```
protected void drawOverlays() {
  mapOverlays = mapView.getOverlays();
  marker = this.getResources().getDrawable(R.drawable.hrpin);
  itemizedOverlay = new CustomItemizedOverlay(marker);
  int latInt = (int)(51.906307 *1E6);
  int lngInt = (int)(4.459666 *1E6);
  GeoPoint gRotterdam = new GeoPoint(latInt, lngInt);
  OverlayItem rotterdam = new OverlayItem(gRotterdam, "Rotterdam", "");

  itemizedOverlay.addOverlay(rotterdam);
  mapOverlays.add(itemizedOverlay);
}
```

mapOverlays zijn alle overlays die op onze mapView staan. Die overlays kunnen heel simpel opgeroepen worden met de functie: `mapView.getOverlays(); .`

```
Met onderstaande functie word de marker opgehaald. In dit geval is de marker de
standaard icoon van deze Applicatie. Voor een marker kan elk type plaatje gebruikt
worden.
marker = this.getResources().getDrawable(R.drawable.icon);
```

De variabele itemizedOverlay is een Object van de CustomItemizedOverlay class die we zojuist gemaakt hebben.

Daarna word er een item aangemaakt die we gaan meegeven aan de ItemizedOverlay. Een overlayItem heeft verplicht een GeoPoint nodig, aangezien zo'n item aan een locatie moet verbonden zijn. Een geopoint is niks anders dan een Lat-Lng omgezet naar een logaritme.

Als laatste moet onze itemizedOverlay nog worden toegevoegd aan de overlays van onze MapView.

Als we alles goed hebben gedaan moet er nu een icoontje op de gewenste plek verschijnen op de map!:)