Upcoming Assignments

- Code Review due Tuesday, February 16 2:10pm
- Pre-alpha version due Wednesday, February 17
- Lab 5 due Monday, February 22
- Read Chapter 7 (Quiz next Friday)
- Read article by Friday (Disaster Mgmt)
- Furlough Day Tuesday, February 16
  - Monday schedule on Tuesday
  - Great time to work with team on Course Project

Maps

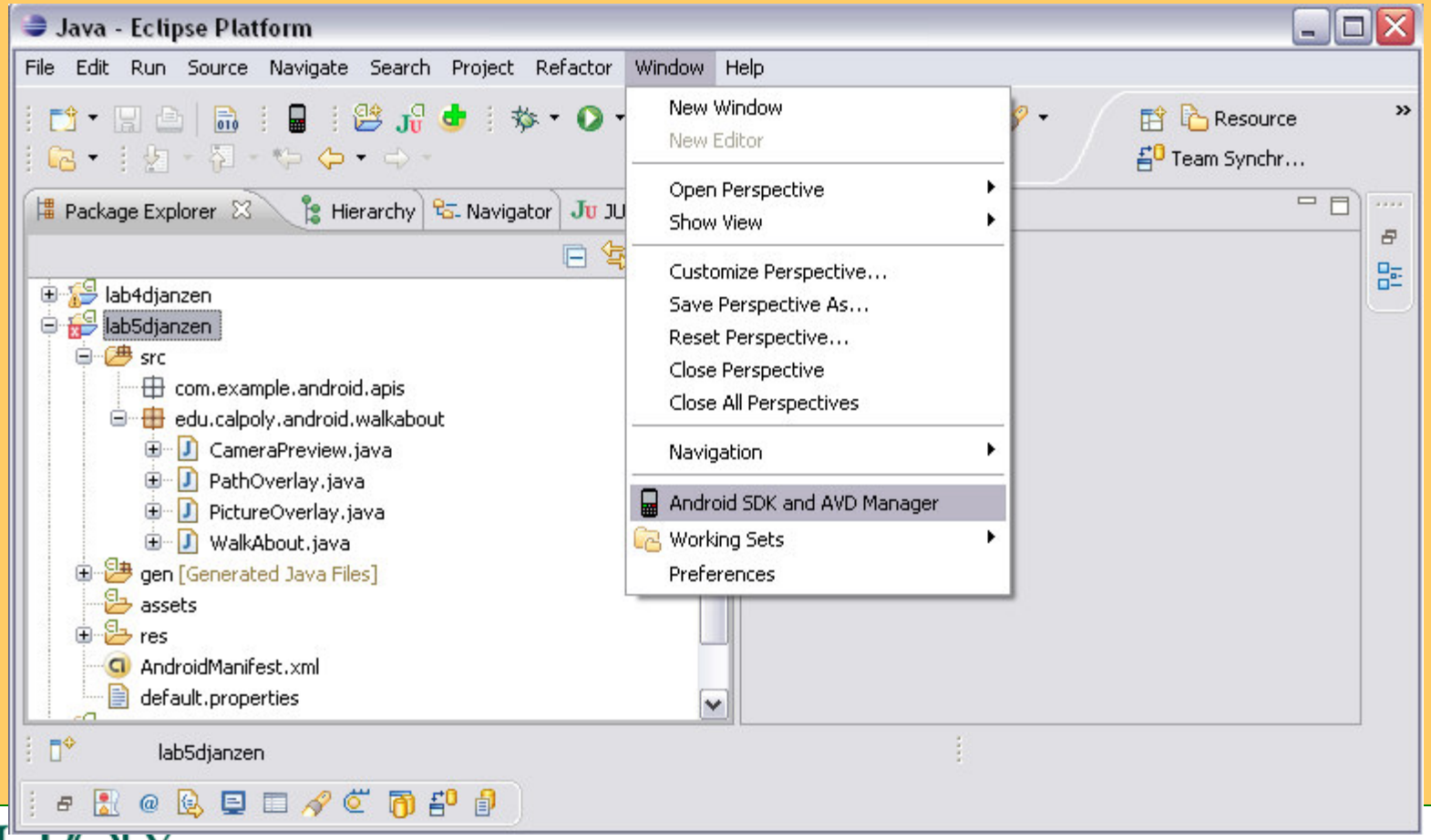
- Google Maps API is widely used on the web
- The Android SDK provides support for easily integrating the Google Maps API

Using Google Maps in our apps

- Configure
  - Maps require the Google API as the project build target
  - Maps require a Map API Key in order to be deployed
  - http://code.google.com/android/add-ons/google-apis/maps-overview.html
- Code
  - Create a MapView in a MapActivity
  - Create Map Overlays

Add Google API in Eclipse

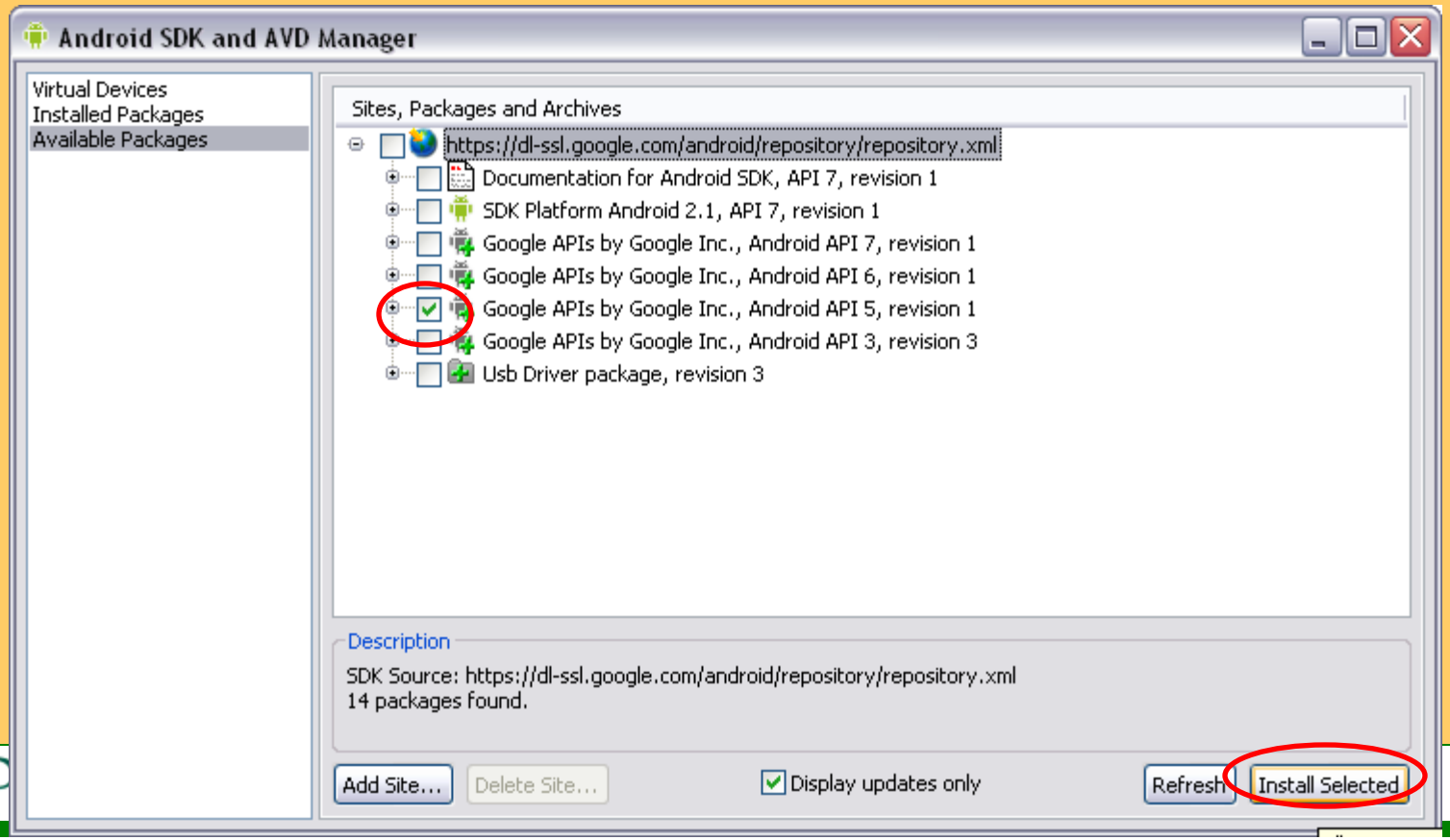
http://developer.android.com/sdk/adding-components.html



Add Google API in Eclipse

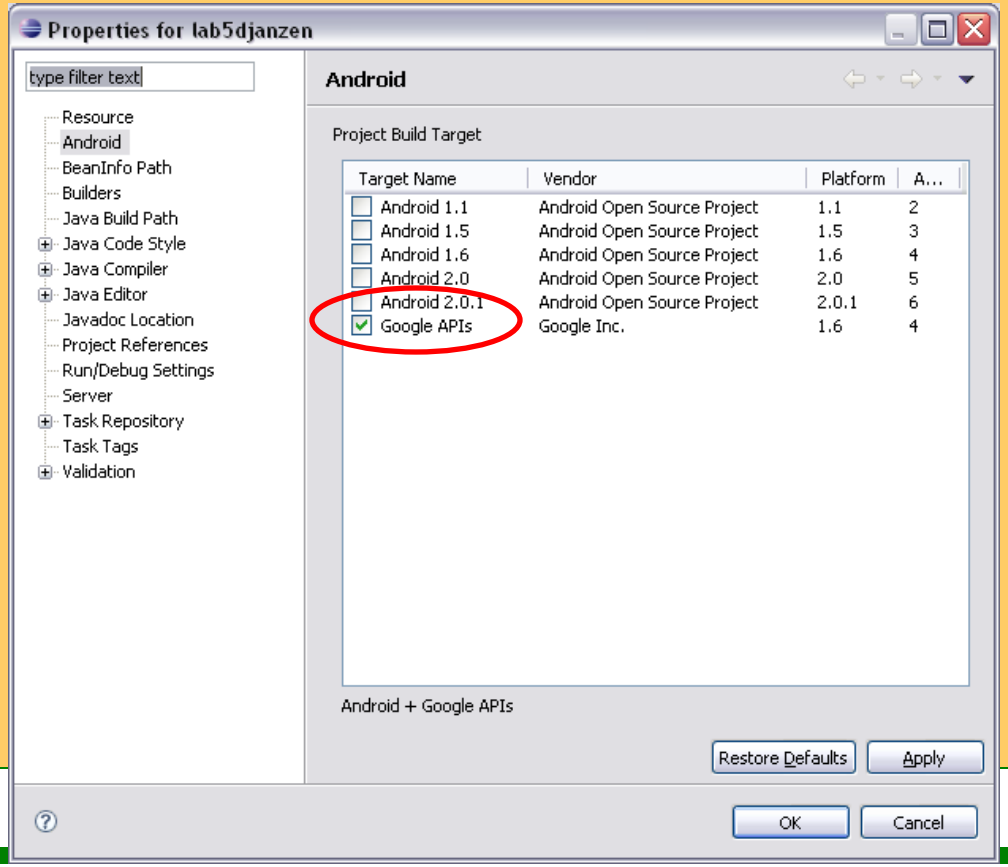
- Use API 4 for SDK 1.6

  http://developer.android.com/guide/appendix/api-levels.html



Add Google API in Eclipse

- Set the Google API as the Project Build Target
  - Right-click on the project, select Properties

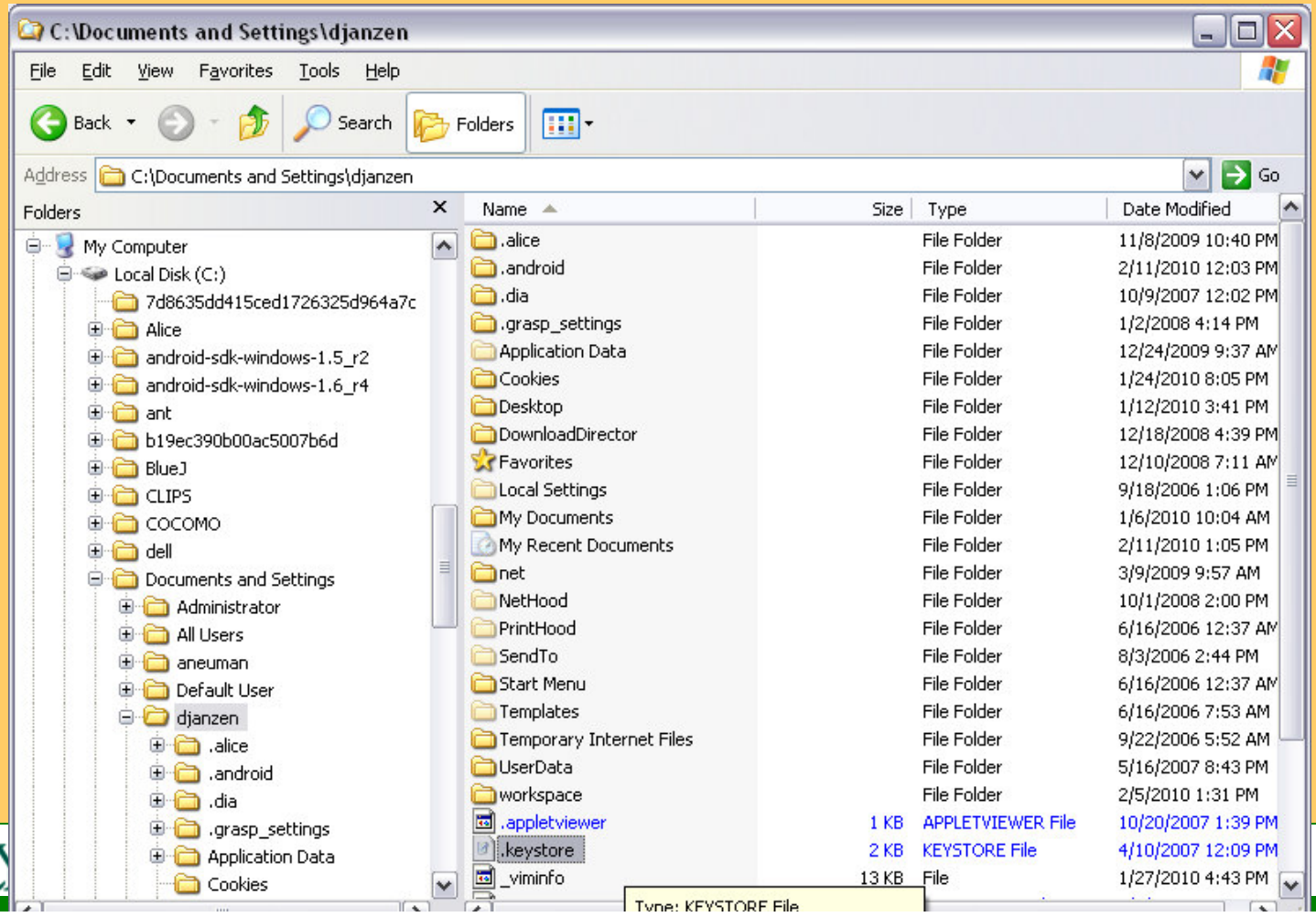


Keys

- As we learned in lab 1 section 6,
  https://sites.google.com/site/androidappcourse/labs/lab-1
  our apps must be signed in order to deploy them on a device
- Eclipse automatically creates a signed debug keystore that is used when launching our app from Eclipse
- In order to deploy our app to the public, we must create a signed keystore
  See http://developer.android.com/guide/publishing/app-signing.html#ExportWizard

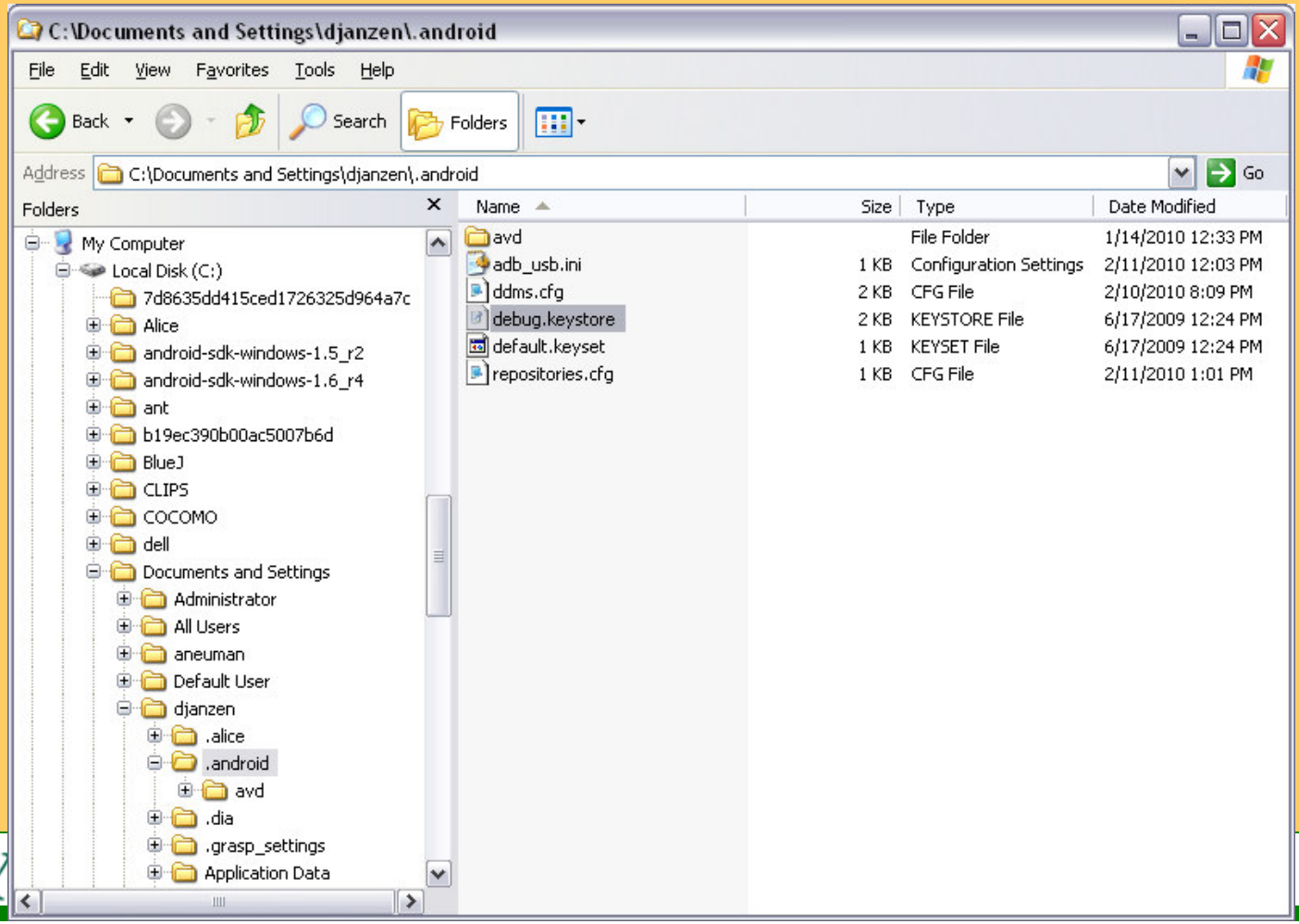
Find your keystore

http://code.google.com/android/add-ons/google-apis/mapkey.html



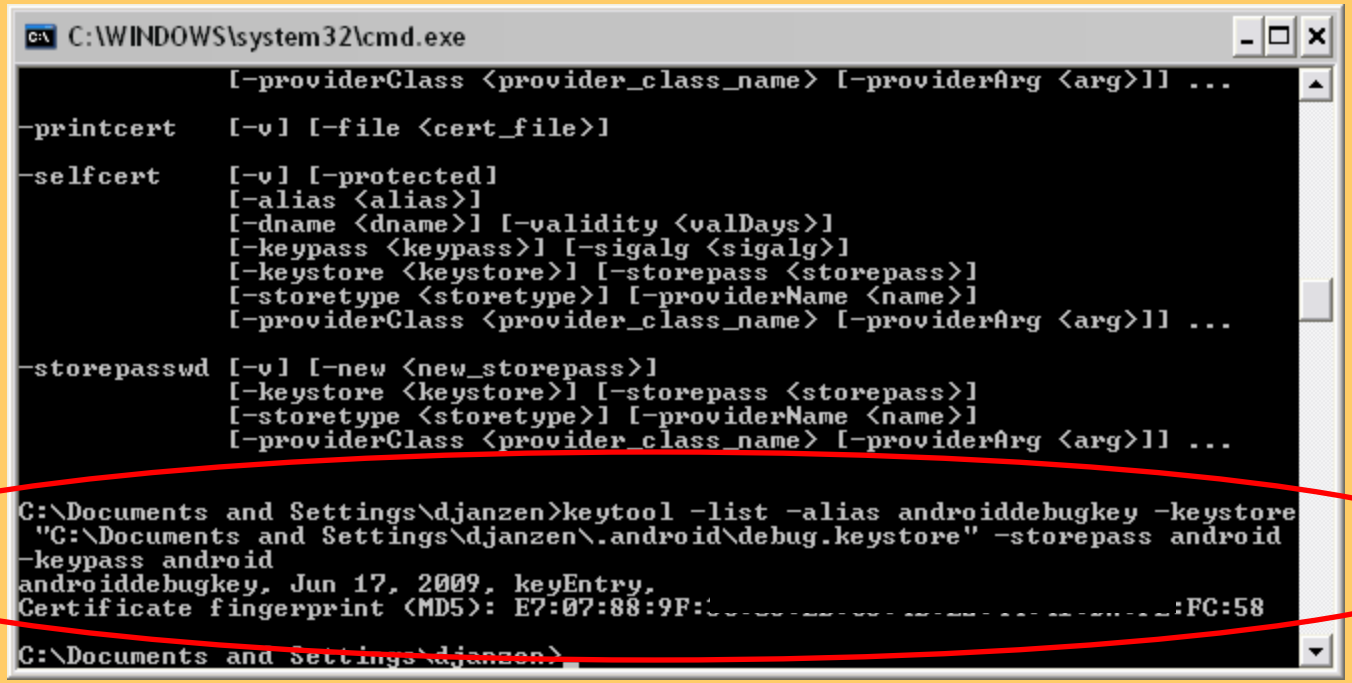
Find your debug keystore

http://code.google.com/android/add-ons/google-apis/mapkey.html



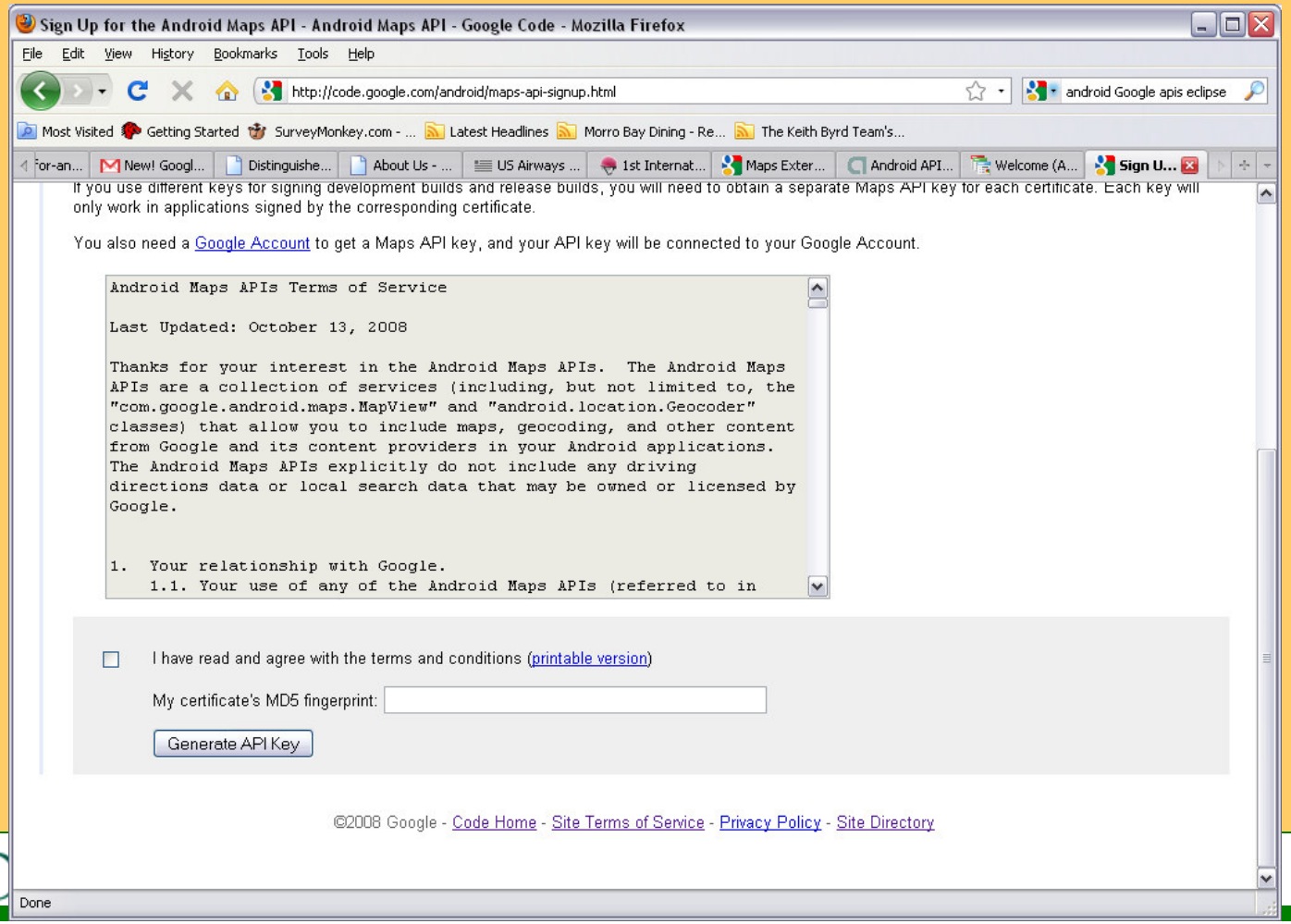
Get your certificate fingerprint

http://code.google.com/android/add-ons/google-apis/mapkey.html



Register your certificate with Google

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



What's in the legal agreement?

- Read the Terms of Service (sections 1-11)
  - http://code.google.com/android/maps-api-signup.html
  - Next quiz may have ?'s on Terms of Service
- Examples
  - Maps may include ads in future
  - Google may limit number of transactions
  - Cannot use for turn-by-turn directions or autonomous driving

Business Risks

- What risks do you take when you use Google Maps in your app?

Add the Map API Key to your Application

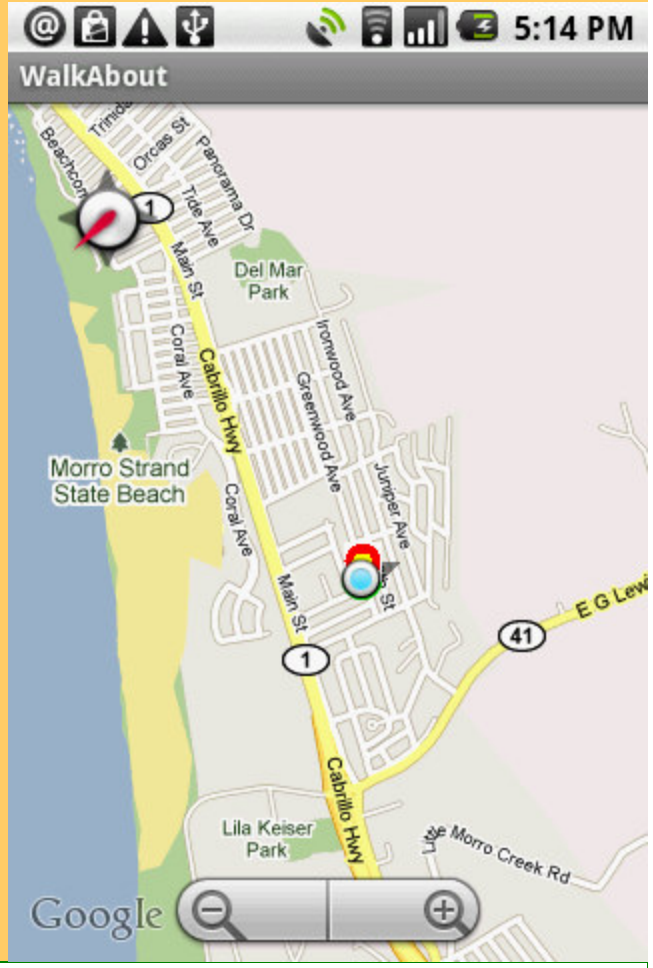
```
<com.google.android.maps.MapView
        android:id="@+id/myMap"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="@string/mapApiKey"/>
```

Configure AndroidManifest.xml

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

Finally, we can start coding

- MapView
  - Contains a map
    - via Google Maps API
    - Map tile retrieval and caching is all done for you
  - Includes pan
  - Includes zoom
    - use setBuiltInZoomControls(true);



MapView Modes

- MapView
  - You determine mode
    - setSatellite(true);
    - setTraffic(true);
    - setStreetView(true);

MapActivity

- MapView can only be constructed or inflated in a MapActivity

```
public class MyActivity extends MapActivity {
…
 @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      …
      MapView myMap = (MapView)findViewById(R.id.myMap);
      myMap.setBuiltInZoomControls();
      myMap.setSatellite(true);
```

MapController

- You can pan and zoom the map programmatically

```
MapView myMap = (MapView)findViewById(R.id.myMap);
MapController mapController = myMap.getController();
mapController.setZoom(1);  //widest zoom/far away
…
mapController.setZoom(21);  //narrowest zoom/close in
mapController.zoomIn();  //one level
mapController.zoomOut();  //one level
```

GeoPoint

- You can move to a particular point

```
MapView myMap = (MapView)findViewById(R.id.myMap);
MapController mapController = myMap.getController();

Double lat = 37.123456 * 1E6;
Double long = -122.123456 * 1E6;
GeoPoint point = new GeoPoint(lat.intValue(), long.intValue());
mapController.setCenter(point);  //jump to point
…
mapController.animateTo(point);  //smooth transition to point
```

Overlays

MyLocationOverlay