# 1. Scope

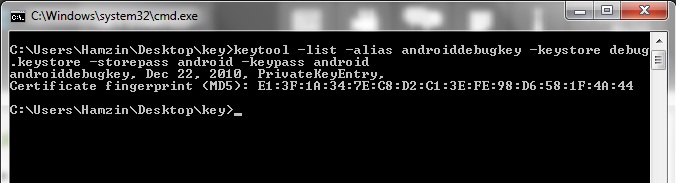
The tutorial will utilize Google Maps API to develop a simple maps application that will allow toggle view between satellite and street views. The application is to use It targets Android 2.1 (Eclair) handsets and assumes **Android 2.1-update1 (API Level 7) or above**. Proceeding from previous labs, it will also continue to strengthen the understanding of event handling and working with permissions.

# 2. Obtaining the Google Maps API Key

It is important to understand at this juncture, any application that utilizes ***’GoogleMaps’*** **requires** a valid **Google Maps API key**. This key can be obtained from, [*http://code.google.com/android/maps-api-signup.html*](http://code.google.com/android/maps-api-signup.html). However in order to obtain this key, it is a pre-requisite to know the md5 value of the **’debug.keystore’**. For this purpose the keytool of java can be utilized.

In order to do this, create a new folder (*‘key’*) on the desktop and copy the ***‘debug.keystore’*** found inside the *.android* folder typically found at ***’Drive:\Users\Your User Name\.android’*** and paste inside the newly created folder (i.e. *‘key’*). Open Command Prompt, change directory to the newly created directory and issue the following command which will return the **MD5 fingerprint** as shown in **Figure1**.

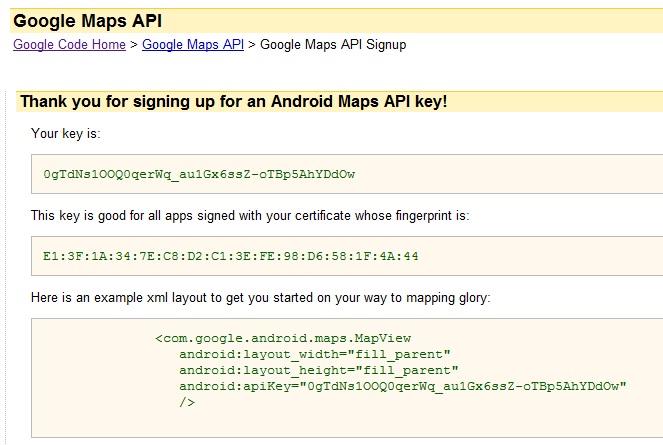
keytool -list -alias androiddebugkey -keystore debug.keystore -storepass android -keypass android



**Figure 1:** Identifying the MD5 fingerprint

**Note:** An example md5 fingerprint will look like, E1:3F:1A:34:7E:C8:D2:C1:3E:FE:98:D6:58:1F:4A:44

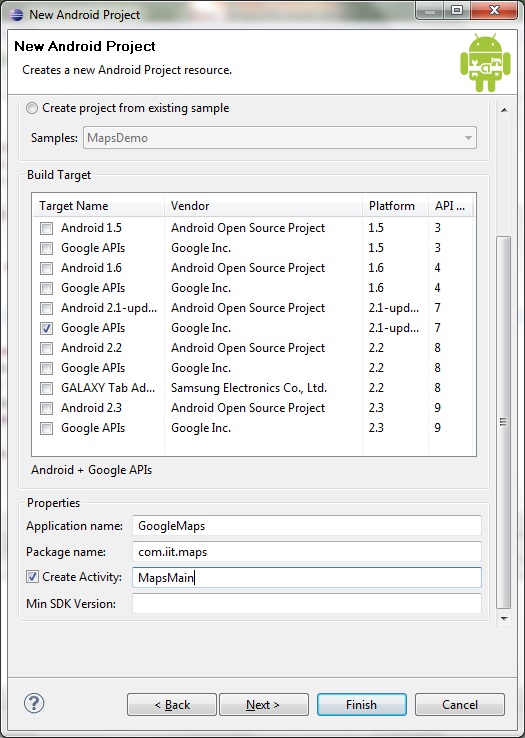
Mark and **copy** the **MD5 Fingerprint** and provide it on, [*http://code.google.com/android/maps-api-signup.html*](http://code.google.com/android/maps-api-signup.html) to **obtain** the **Google Maps API Key** as shown in **Figure2**.



**Figure 2:** Obtaining a Google Maps API key

# 3. Create the Android Project

Create a new Android Project (**File -> New -> Other -> Android Project**). Click **Next** on the first dialog and continue to fill out project details. For project name and application name provide ‘***GoogleMaps***’ and for package name ‘***com.iit.maps***’. Name the main activity as ‘***MapsMain***’ and select a suitable **GoogleAPIs** as the as the build target. Rest of the configurations can be left as default as shown on **Figure3**.

****

**Figure 3:** Project Details

# 4. The Layout

Unlike in the previous labs, this application will use a relative layout. The xml source of the layout can be seen below. It is **important** to provide the **Google Maps API Key** obtained **for** ***‘android:apiKey’*** property.

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"* android:layout\_height=*"fill\_parent"*>

<com.google.android.maps.MapView

xmlns:android=*"http://schemas.android.com/apk/res/android"* android:id=*"@+id/mapView"*

android:layout\_width=*"fill\_parent"* android:layout\_height=*"fill\_parent"*

android:clickable=*"true"* android:enabled=*"true"*

android:apiKey=*"0gTdNs1OOQ0qerWq\_au1Gx6ssZ-oTBp5AhYDdOw"* />

<LinearLayout android:id=*"@+id/zoom"* android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* android:layout\_alignParentBottom=*"true"*

android:layout\_centerHorizontal=*"true"* />

</RelativeLayout>

# 5. Creating the Menu

Create a menu folder named ***‘menu’*** under res folder and create a menu file named ***‘maps\_menu.xml’*** to hold two menu items, ***‘Toggle View’*** and ***‘Close’***.

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<item android:id=*"@+id/item01"* android:title=*"Toggle View"*></item>

<item android:id=*"@+id/item02"* android:title=*"Close"*></item>

</menu>

# 6. Implementing the Map Activity

The code for the ‘***MapsMain***’ activity can be seen below. It creates a maps overlay and sets its target to "6.865164", "79.859874" (i.e. latitude and longitude of IIT). In addition to the built-in zooming provided with pinch gesture, the activity utilizes the ‘***zoomController***’ to allow the same functionality via **buttons** *(zoom-in and zoom-out)* that will be displayed at the bottom of the activity.

**package** com.iit.maps;

**import** com.google.android.maps.GeoPoint;

**import** com.google.android.maps.MapActivity;

**import** com.google.android.maps.MapController;

**import** com.google.android.maps.MapView;

**import** android.os.Bundle;

**import** com.google.android.maps.MapView.LayoutParams;

**import** android.view.KeyEvent;

**import** android.view.Menu;

**import** android.view.MenuInflater;

**import** android.view.MenuItem;

**import** android.view.MotionEvent;

**import** android.view.View;

**import** android.widget.LinearLayout;

**public** **class** MapsMain **extends** MapActivity {

MapView mapView;

MapController mc;

GeoPoint p;

**private** **static** **int** *flag\_view* = 0;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

mapView = (MapView) findViewById(R.id.*mapView*);

LinearLayout zoomLayout = (LinearLayout) findViewById(R.id.*zoom*);

View zoomView = mapView.~~getZoomControls~~();

zoomLayout.addView(zoomView, **new** LinearLayout.LayoutParams(

LayoutParams.*WRAP\_CONTENT*, LayoutParams.*WRAP\_CONTENT*));

mapView.displayZoomControls(**true**);

// displaying a particular point

mc = mapView.getController();

String coordinates[] = { "6.865164", "79.859874" };

**double** lat = Double.*parseDouble*(coordinates[0]);

**double** lng = Double.*parseDouble*(coordinates[1]);

p = **new** GeoPoint((**int**) (lat \* 1E6), (**int**) (lng \* 1E6));

mc.animateTo(p);

mc.setZoom(18);

mapView.invalidate();

}

@Override

**protected** **boolean** isRouteDisplayed() {

// **TODO** Auto-generated method stub

**return** **false**;

}

**public** **boolean** onKeyDown(**int** keyCode, KeyEvent event) {

MapController mc = mapView.getController();

**switch** (keyCode) {

**case** KeyEvent.*KEYCODE\_3*:

mc.zoomIn();

**break**;

**case** KeyEvent.*KEYCODE\_1*:

mc.zoomOut();

**break**;

}

**return** **super**.onKeyDown(keyCode, event);

}

}

# 7. Handling the Menu Items

The code below is used to handle the menu items to allow toggling view when the toggle menu item is pressed and to quit the application when the Close menu item Is pressed.

**public** **boolean** onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.*maps\_menu*, menu);

**return** **true**;

}

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.*item01*:

// currently shows map view(default) [show satellite]

**if** (*flag\_view* == 0) {

mapView.setSatellite(**true**);

*flag\_view* = 1;

} **else** {

// currently shows satellite view [show street]

mapView.setSatellite(**false**);

*flag\_view* = 0;

}

**break**;

**case** R.id.*item02*:

// close the application

finish();

**break**;

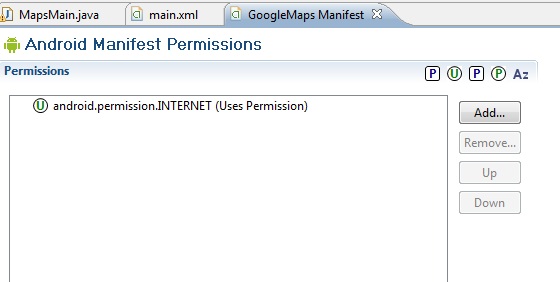
}

**return** **true**;

}

# 8. Permissions

The application requires a special permission to access **internet**. This can be provided on the ***AndroidManifest*** under its ***Permissions*** perspective as shown in **Figure4**.



**Figure 4:** The permission required for the application

# 10. Screen shots

**Figure 5:** Screen shots of the application