­­School of Digital Media and Infocomm Technology (DMIT)

ST0281 Mobile Applications

Practical 17

|  |
| --- |
| Objectives:  After completing this lab, you will be able to:   * Use LocationManager and LocationListener classes in Android * Use Map API version 2 * Display maps in your Android application |

Part A

In this part of the practical, you will create a new Android project to implement an application to make use of the GPS location provider to obtain your location and then display the latitude and longitude coordinates in a Toast message.

Step 1:

Create a new project called MyLocationGPS.

|  |  |
| --- | --- |
| 1. Application name: MyLocationGPS | 2.  p8_2 |
| 3. Empty Activity  p8_3 | 4. Activity Name:MainActivityGPS |

Step 2:

Create a new Java class called LocationTracker and add in the code as shown:

|  |
| --- |
| package mapp.com.sg.mylocationgps;  import android.Manifest;  import android.app.AlertDialog;  import android.app.Service;  import android.content.Context;  import android.content.DialogInterface;  import android.content.Intent;  import android.location.Location;  import android.location.LocationListener;  import android.location.LocationManager;  import android.os.Bundle;  import android.os.IBinder;  import android.provider.Settings;  import android.util.Log;  /\*\*  \* Created by s38092  \*/  public class LocationTracker extends Service implements LocationListener {  private final Context mContext;  // flag for GPS status  boolean isGPSEnabled = false;  // flag for network status  boolean isNetworkEnabled = false;  // flag for GPS status  boolean canGetLocation = false;  Location location; // location  double latitude; // latitude  double longitude; // longitude  // The minimum distance to change Updates in meters  private static final long MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES = 10; // 10 meters  // The minimum time between updates in milliseconds  private static final long MIN\_TIME\_BW\_UPDATES = 1000 \* 60 \* 1; // 1 minute  // Declaring a Location Manager  protected LocationManager locationManager;  private static final int REQUEST\_CODE\_PERMISSION = 2;  String mPermission = Manifest.permission.ACCESS\_FINE\_LOCATION;  public LocationTracker(Context context) {  this.mContext = context;  getLocation();  }  public Location getLocation() {  try {  locationManager = (LocationManager) mContext.getSystemService(LOCATION\_SERVICE);  // getting GPS status  isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER);  // getting network status  isNetworkEnabled = locationManager  .isProviderEnabled(LocationManager.NETWORK\_PROVIDER);  if (!isGPSEnabled && !isNetworkEnabled) {  // no network provider is enabled  } else {  this.canGetLocation = true;  // First get location from Network Provider  if (isNetworkEnabled) {  /\* if (ActivityCompat.checkSelfPermission(this,  // Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED  // && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  // TODO: Consider calling  // ActivityCompat#requestPermissions  // here to request the missing permissions, and then overriding  // public void onRequestPermissionsResult(int requestCode, String[] permissions,  // int[] grantResults)  // to handle the case where the user grants the permission. See the documentation  // for ActivityCompat#requestPermissions for more details.  //return ;  Already check in the launching  } \*/  locationManager.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER,  MIN\_TIME\_BW\_UPDATES, MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);  Log.d("Network", "Network");  if (locationManager != null) {  location = locationManager.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);  if (location != null) {  latitude = location.getLatitude();  longitude = location.getLongitude();  }  }  }  // if GPS Enabled get lat/long using GPS Services  if (isGPSEnabled) {  if (location == null) {  locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER,  MIN\_TIME\_BW\_UPDATES, MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);  Log.d("GPS Enabled", "GPS Enabled");  if (locationManager != null) {  location = locationManager.getLastKnownLocation(LocationManager.GPS\_PROVIDER);  if (location != null) {  latitude = location.getLatitude();  longitude = location.getLongitude();  }  }  }  }  }  } catch (Exception e) {  e.printStackTrace();  }  return location;  }  /\*\*  \* Stop using GPS listener  \* Calling this function will stop using GPS in your app  \* \*/  public void stopUsingGPS() {  if (locationManager != null) {  /\* if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  // TODO: Consider calling  // ActivityCompat#requestPermissions  // here to request the missing permissions, and then overriding  // public void onRequestPermissionsResult(int requestCode, String[] permissions,  // int[] grantResults)  // to handle the case where the user grants the permission. See the documentation  // for ActivityCompat#requestPermissions for more details.  //return;  // Already check in the launching  }\*/  locationManager.removeUpdates(LocationTracker.this);  }  }  /\*\*  \* Function to get latitude  \* \*/  public double getLatitude(){  if(location != null){  latitude = location.getLatitude();  }  // return latitude  return latitude;  }  /\*\*  \* Function to get longitude  \* \*/  public double getLongitude(){  if(location != null){  longitude = location.getLongitude();  }  // return longitude  return longitude;  }  /\*\*  \* Function to check GPS/wifi enabled  \* @return boolean  \* \*/  public boolean canGetLocation() {  return this.canGetLocation;  }  /\*\*  \* Function to show settings alert dialog  \* On pressing Settings button will lauch Settings Options  \* \*/  public void showSettingsAlert(){  AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);  // Setting Dialog Title  alertDialog.setTitle("GPS is settings");  // Setting Dialog Message  alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");  // On pressing Settings button  alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog,int which) {  Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);  mContext.startActivity(intent);  }  });  // on pressing cancel button  alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int which) {  dialog.cancel();  }  });  // Showing Alert Message  alertDialog.show();  }  @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) {  }  @Override  public IBinder onBind(Intent arg0) {  return null;  }  } |

Step 3:

Modify the AndroidManifest file.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="mapp.com.sg.mylocationgps">  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  <uses-permission android:name="android.permission.INTERNET" />  <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:supportsRtl="true"  android:theme="@style/AppTheme">  <activity android:name=".MainActivityGPS">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

Step 4:

Modify the MainActivityGPS.java files as follows:

|  |
| --- |
| package mapp.com.sg.mylocationgps;  import android.Manifest;  import android.support.v4.app.ActivityCompat;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.test.mock.MockPackageManager;  import android.view.View;  import android.widget.Button;  import android.widget.Toast;  public class MainActivityGPS extends AppCompatActivity {  Button btnShowLocation;  private static final int REQUEST\_CODE\_PERMISSION = 2;  String mPermission = Manifest.permission.ACCESS\_FINE\_LOCATION;  // LocationTracker class  LocationTracker gps;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main\_gps);  try {  if (ActivityCompat.checkSelfPermission(this, mPermission)  != MockPackageManager.PERMISSION\_GRANTED) {  ActivityCompat.requestPermissions(this, new String[]{mPermission},  REQUEST\_CODE\_PERMISSION);  // If any permission above not allowed by user, this condition will  // execute every time, else your else part will work  }  } catch (Exception e) {  e.printStackTrace();  }  btnShowLocation = (Button) findViewById(R.id.button);  // show location button click event  btnShowLocation.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View arg0) {  // create class object  gps = new LocationTracker(MainActivityGPS.this);  // check if GPS enabled  if(gps.canGetLocation()){  double latitude = gps.getLatitude();  double longitude = gps.getLongitude();  // \n is for new line  Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "  + latitude + "\nLong: " + longitude, Toast.LENGTH\_LONG).show();  }else{  // can't get location  // GPS or Network is not enabled  // Ask user to enable GPS/network in settings  gps.showSettingsAlert();  }  }  });  }  } |

Step 5:

Save all files and run the application.

Click on the GETLOCATION button to display your current location.

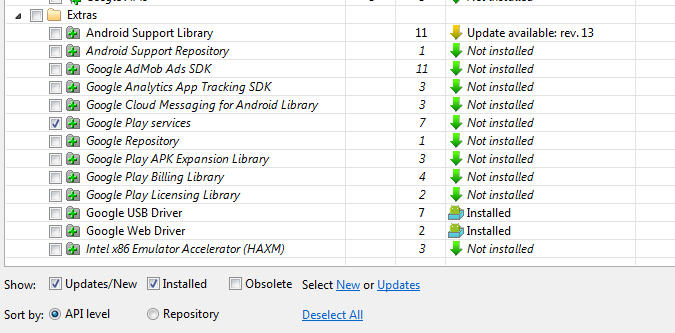
A toast will appear with the latitude and longitude of your location.

|  |  |
| --- | --- |
|  |  |

Part B

In this part you will learn how to use Google Map API version 2.

First, download the Google Play services library using the Android SDK Manager. The Google Play services library is located as shown:

****

The downloaded Google Play services library will be in the following location:

*<android-sdk-folder>/extras/google/google\_play\_services/libproject/google-play-services\_lib/*

Displaying maps in your application using MapFragment

The google play services library needs to be inside your project if you want to show maps in your application.

Step 1:

Create a new project called MapExample.

|  |  |
| --- | --- |
| 1. Application name: MapExample | 2.  p8_2 |
| 3Google Map Activity | 4. Activity Name:MapActivity |

Once you have created your new project in Android, select File->Project Structure”.

Then add the google play service library into your project:

After clicking the add sign, select *Library Dependency* and browse to the package “com.google.android.gms:play-services:10.0.1”. Note that the version depends on what version of the Google Play Services you have installed in the SDK Manager.

Click “OK”.

You can check the *build.gradle* file under your “app” to see that the dependency has been added.

|  |
| --- |
|  |

Part C:

Create your own credential for the API key.

Step 1:

In your web browser, go to https://console.developers.google.com/ and log in using your own google account. You should then see this screen:

Note: You need to have google account to generate the key.

|  |
| --- |
|  |

Step 2:

Click on the “Create project’ blue button and you will see the following screen:

Project name: My Map Projects

|  |
| --- |
|  |

Step 3:

Select Library.

|  |
| --- |
|  |

Step 4:

Search for google map android api.

Select Google Maps Andorid API.

|  |
| --- |
|  |

Step 5:

Click the blue ENABLE to enable the Google Maps Android API.

|  |
| --- |
|  |

Step 6:

Display as shown when it is ENABLE.

Create on the Create credentials (blue button). To create an API key.

|  |
| --- |
|  |

Step 7:

The API key will generated as shown:

Use this API key for your Google Map project.

|  |
| --- |
|  |

Click on the yellow exclamation mark, to restrict the use of the API key.

|  |
| --- |
|  |

Step 8:

Select the Android apps.

|  |
| --- |
|  |

In this part you will learn how to obtain the Google Map API key to be used in your app so that your app can display maps.

To obtain the Google Map API key, you need to use the keytool command (located at C:\Program Files\Java\jdk1.7.x\bin (check your own path of your JDK installation)) to generate the MD5/SHA1 certificate fingerprint of the debug.keystore file. This file is located at the .android folder. Eclipse uses this file to sign your application whenever you test run it on the emulator or actual device.

First, make sure you have a path to the keytool.exe file.

Note: You may SetPath using command prompt or follow the following step.

Step 9:

Run -> cmd.

Using cd command to go to the keytool command in the Java directory.

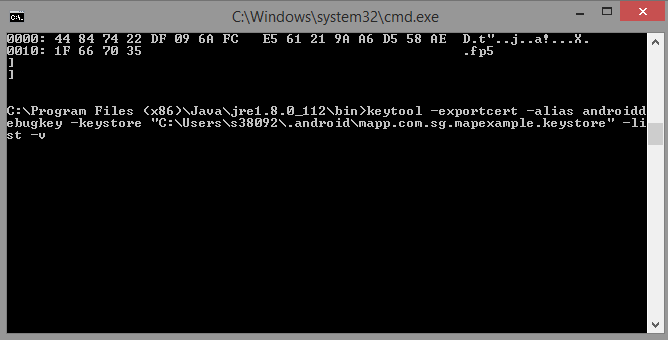
C:\program files (x86)\java\jre1.8.0\_112\bin

Once the keytool command directory.

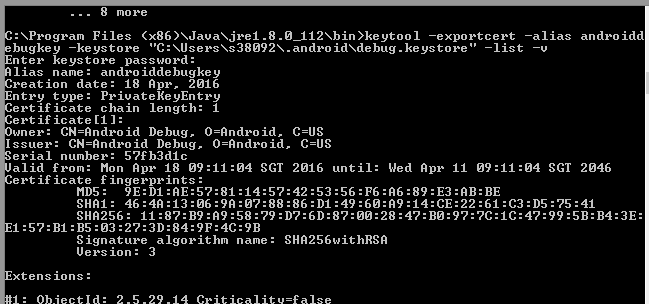
Enter the following command:

keytool -exportcert -alias androiddebugkey -keystore "C:\Users\userName\.android\debug.keystore" -list -v

where userName = your directory user name.



Generated the SHA1.



Step 9:

Enter the SHA1 into the SHA1 certificate fingerprint.

|  |
| --- |
|  |

Part D:

Now that you have the Map API key, you can go back to your new Android project to display a map.

Step 1:

Modify the AndroidManifest file to include the permission to access the internet and the use of the Google map library:

|  |
| --- |
| <? <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="mapp.com.sg.mapexample">  <!--  The ACCESS\_COARSE/FINE\_LOCATION permissions are not required to use  Google Maps Android API v2, but you must specify either coarse or fine  location permissions for the 'MyLocation' functionality.  -->  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:supportsRtl="true"  android:theme="@style/AppTheme">  <!--  The API key for Google Maps-based APIs is defined as a string resource.  (See the file "res/values/google\_maps\_api.xml").  Note that the API key is linked to the encryption key used to sign the APK.  You need a different API key for each encryption key, including the release key that is used to  sign the APK for publishing.  You can define the keys for the debug and release targets in src/debug/ and src/release/.  -->  <meta-data  android:name="com.google.android.geo.API\_KEY"  android:value="YOUR Key" />  <activity  android:name=".MapsActivity"  android:label="@string/title\_activity\_maps">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

Step 2:

Next, modify the MapsActivity.java file to display the location in Singapore.

|  |
| --- |
| @Override  public void onMapReady(GoogleMap googleMap) {  mMap = googleMap;  // Add a marker in Sydney and move the camera  LatLng sg = new LatLng(1.308349, 103.819836);  mMap.addMarker(new MarkerOptions().position(sg).title("Marker in Singapore"));  mMap.moveCamera(CameraUpdateFactory.newLatLng(sg));  } |

Step 3:

Run the application using a real Android device (version 4 and above), you should see the map appearing:

|  |
| --- |
|  |

Note the above map will run in Android version 4 and above. In order for older versions to support the fragment widget, you will need to make use of the Android Support Package.

Tasks

Combine Part A with Part B to be able to locate your current location.