Setup Google Map API Project in Android Studio (from Template)

Part 2: Setup Google Map API Project in Android Studio

Everytime when you create a map app, please do following steps. Before starting, please make sure the location of your Android Studio installation directory.

For example

- Mac: Applications > Android Studio.app > Right Click > Show Package Contents
- Windows: C:\Users\Your Username\AppData\Local\Android\android-studio\

(**AppData** is a hidden folder. To unhide hidden folders, go to **My Computer**, click on **Alt** key on your keyboard, click on **Tools** from the menu bar, **Folder Options...**, Click on **View** tab, click on the radio button, **Show hidden files, folders, and drives** and click **OK**.)

Also, know your Android Studio Projects folder. Usually, it is under your user's home directory.

Overview

1. Copy the template project and rename
2. Setup the dependency
3. Setup the Google API key
4. Edit the build configurations

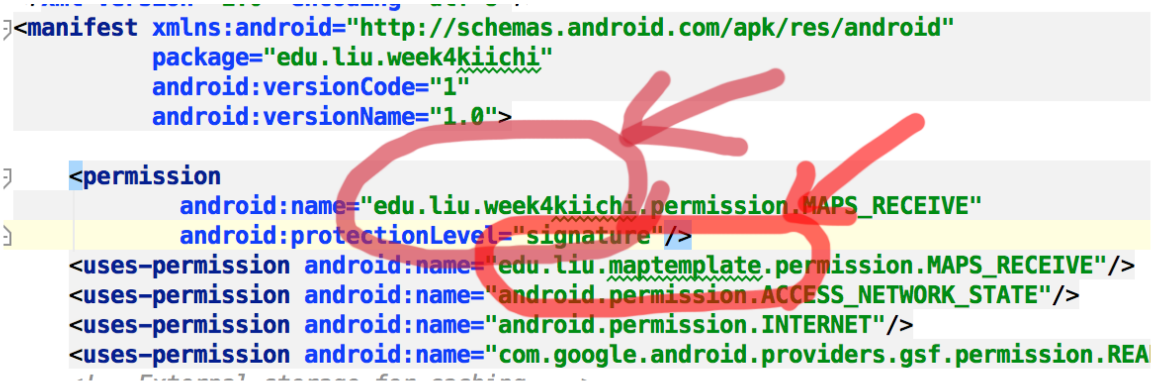
Steps

1. Download the template project from here https://www.dropbox.com/s/1p1e9nj3kwoglb9/MyMapTemplate.zip
2. Extract the zip file.
3. We are going to clone and rename this template project.
4. Copy the folder into your Android Project folder.
5. Rename the folder to your app name. In this example, we will create an app, "HelloMap".
6. Start Android Studio. Close if there is any previous open project.
7. In the first screen, click Import Project.
8. Select the folder that you just cloned.
9. Click next and finish the wizard.
10. Select Refactor > Rename. Type your new app name. In this case, we use HelloMap.
11. Expand the folder > src, and right click on edu.liu.mymaptemplate
12. Select Refactor > Rename. Select Rename Package.

13. Type your new app name in lowercase (e.g. hellomap). This is handy tool to rename all related files in one shot.

14. Click Refactor and finish this process.

15. Sometimes it forgets to change the permission in AndroidManifest.xml. It often forget to rename the package name in uses-permission line. Double check the file.



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="edu.liu.week4kiichi"
        android:versionCode="1"
        android:versionName="1.0">

    <permission
            android:name="edu.liu.week4kiichi.permission.MAPS_RECEIVE"
            android:protectionLevel="signature"/>
    <uses-permission android:name="edu.liu.maptemplate.permission.MAPS_RECEIVE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.REA
```

16. In next few steps, we will add Google Play Services Library as the dependency of this project because Google Map API is included in it.

17. From the menu, File > Project Structure > Modules > Click Plus (+) Icon > Import Module > Go to your Android Studio directory, and then find following path sdk/extras/google/google_play_services/libproject/google-play-services_lib

18. Select **Create Module from Existing Sources**

19. Click next to finish. If it asks you to overwrite setting file, just overwrite it.

20. In the same Modules tab, select the name of your app (e.g. MyMap).

21. Click Plus (+) Icon at the right side pane > Module Dependency > google-play-services_lib

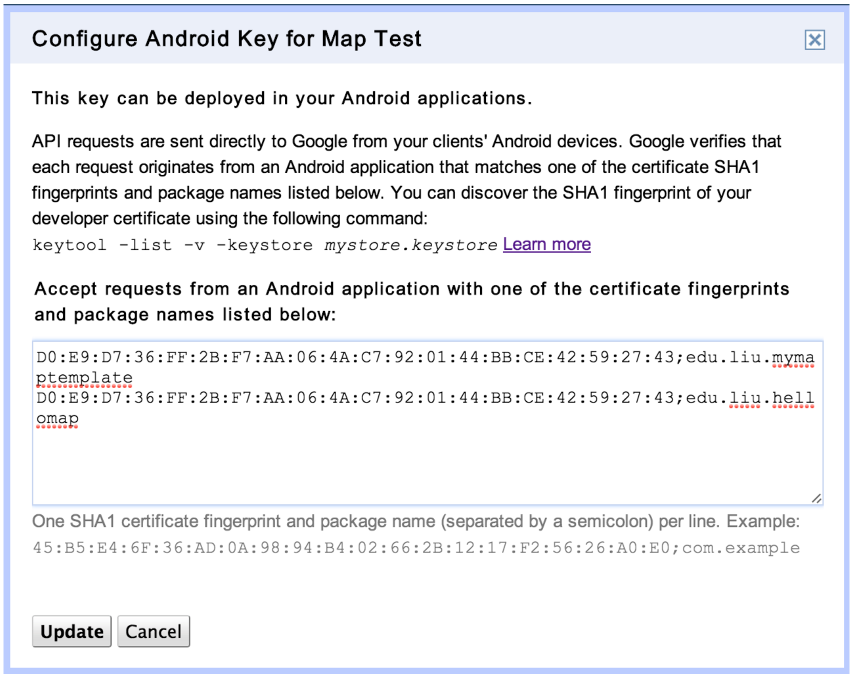
22. Click Plus (+) Icon at the right side pane > Jars or directories > Go to your Android Studio directory (sdk folder), and then find following path for the .jar file sdk/extras/google/google_play_services/libproject/google-play-services_lib/libs/**google-play-services.jar**

23. Use the up/down arrows to move <Module source> to the bottom of the list.

24. Click OK and close the window.

25. Before deploy this app, we need to setup Google Map API Key on both **Server Side (Google API Console) and Client Side (Android Studio)**.

26. Add a new Google Map API Console's key. You need to add a new line as you create a new app. Go to Google API Console website > API Access > Click Edit. Replicate the first line, and you will replace the last part with your new app's package name (e.g. edu.liu.hellomap)



This key can be deployed in your Android applications.

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:
`keytool -list -v -keystore mystore.keystore` Learn more

Accept requests from an Android application with one of the certificate fingerprints and package names listed below:

```
D0:E9:D7:36:FF:2B:F7:AA:06:4A:C7:92:01:44:BB:CE:42:59:27:43;edu.liu.mymaptemplate
D0:E9:D7:36:FF:2B:F7:AA:06:4A:C7:92:01:44:BB:CE:42:59:27:43;edu.liu.hellomap
```

One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:
`45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example`

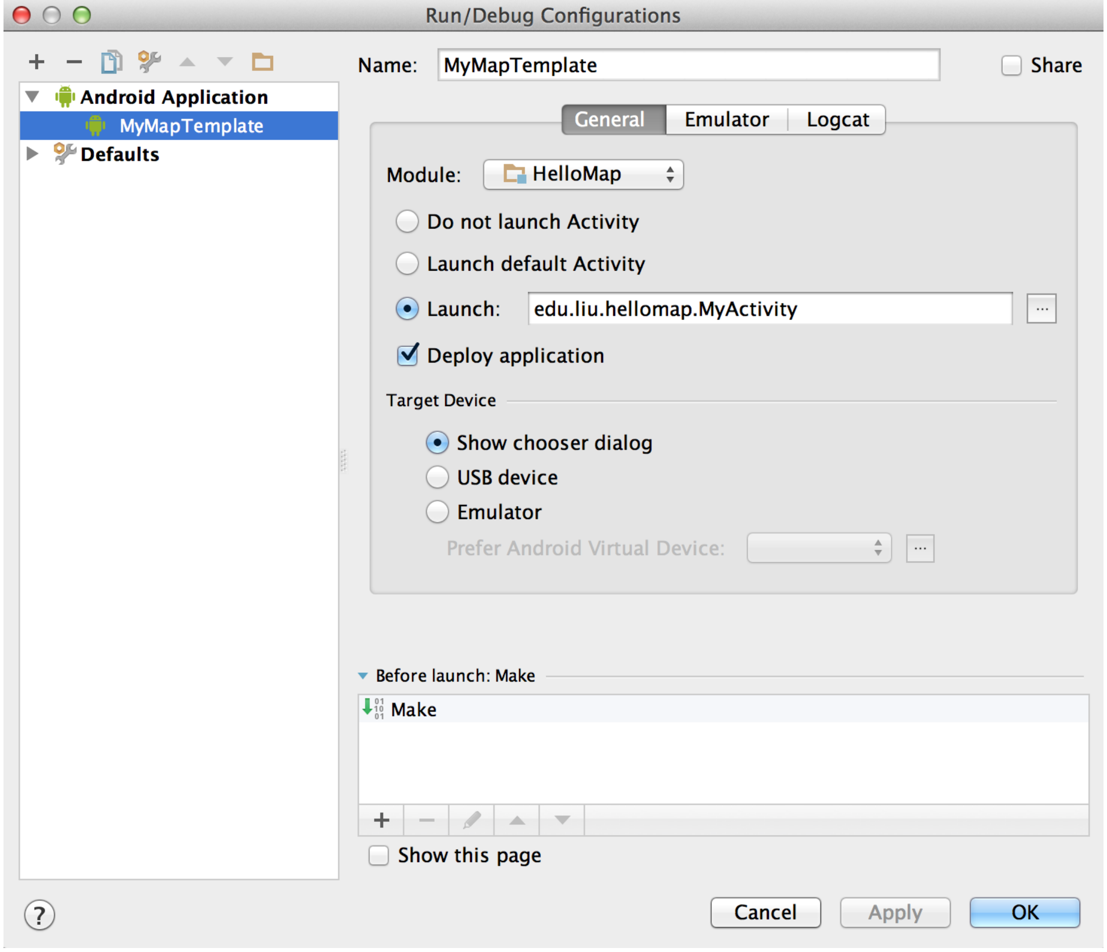
27. Click Update, and copy the API Key.
28. Open AndroidManifest.xml, and replace the key in the meta-data tag.

```
<meta-data
             android:name="com.google.android.maps.v2.API_KEY"
             android:value="YOUR_API_KEY" />
```

Also, double check your AndroidManifest.xml does not contain the previous package name. There are a few places change (package name and permissions)

```
<permission
        android:name="edu.liu.hellomap.permission.MAPS_RECEIVE"
        android:protectionLevel="signature"/>
 <uses-permission android:name="edu.liu.hellomap.permission.MAPS_RECEIVE"/>
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
 <uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
 <!-- External storage for caching. -->
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
 <!-- My Location -->
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
 <!-- Maps API needs OpenGL ES 2.0. -->
 <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

29. Click the dropdown next to the play button. Select Edit Configurations.
30. Select your app name in the Module dropdown item.
31. fix the Launch package name (e.g. edu.liu.hellomap.MyActivity).
32. Select "Show chooser dialog". The configurations should look like below.



Name: MyMapTemplate

□ Share

Module: HelloMap

○ Do not launch Activity

○ Launch default Activity

● Launch: edu.liu.hellomap.MyActivity

☑ Deploy application

Target Device

● Show chooser dialog

○ USB device

○ Emulator

Prefer Android Virtual Device:

□ Show this page

33. Click OK to close the window.

34. Click play button, and select your device to deploy. Make sure the base map is appearing, otherwise, you must have problem with API Key settings in previous steps. Additionally, make sure your device has internet connection.

Appendix: Implement Google Map API from Scratch

At this point, Android Studio does not work well to setup the map project from the scratch, and we took the approach to use a template project. However, in the future when they fix those issues, or in case you need to set it up in Eclipse or IntelliJ CE, here is the instruction.

1. First, we need to add permissions, and Google Map API Key in the manifest file. Double click and open **AndroidManifest.xml** in your App's project folder > Your app name > src folder.
2. Add following permissions After <manifest ……. > tag. Please note that **change the name with your app's package name**. See the red font lines. This is case sensitive, and if you forgot it, look at your manifest tag's package attribute.

```
<permission
         android:name="edu.liu.mymap.permission.MAPS_RECEIVE"
         android:protectionLevel="signature"/>
 <uses-permission android:name="edu.liu.mymap.permission.MAPS_RECEIVE"/>
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
 <uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
 <!-- External storage for caching. -->
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
 <!-- My Location -->
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
 <!-- Maps API needs OpenGL ES 2.0. -->
 <uses-feature
         android:glEsVersion="0x00020000"
         android:required="true"/>
```

3. After <application ….. > tag, add your Google API Key. Paste following lines. Make sure to use your own key from the previous steps.

```
    <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="YOUR_API_KEY_GOES_HERE" />
```

4. The final manifest file should look like this. I highlighted changed locations in yellow color.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="edu.liu.testm3"
   android:versionCode="1"
   android:versionName="1.0" >

    <permission
           android:name="edu.liu.mymap.permission.MAPS_RECEIVE"
           android:protectionLevel="signature"/>
   <uses-permission android:name="edu.liu.mymap.permission.MAPS_RECEIVE"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-
permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
   <!-- External storage for caching. -->
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <!-- My Location -->
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
   <!-- Maps API needs OpenGL ES 2.0. -->
   <uses-feature
           android:glEsVersion="0x00020000"
           android:required="true"/>

   <uses-sdk
       android:minSdkVersion="8"
       android:targetSdkVersion="16" />

   <application
       android:allowBackup="true"
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >


        <meta-data
               android:name="com.google.android.maps.v2.API_KEY"
               android:value="YOUR_API_KEY" />


       <activity
           android:name="edu.liu.testm3.MainActivity"
           android:label="@string/app_name" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
   </application>
</manifest>
```

5. Save the manifest file and close the tab.

6. In the project, we will add Google Map API in the layout.

7. Open Your App Project > Your App Folder > src > main > res > layout > activity_main.xml

8. Paste the following codes. We are using a new component, it's called "Fragment". This is subset of activity and we will discuss more details about this class later.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
       >
   <fragment
           android:id="@+id/map"
           android:name="com.google.android.gms.maps.MapFragment"
           android:layout_width="match_parent"
           android:layout_height="match_parent" />
</LinearLayout>
```

9. Quit Android Studio and Restart. If there is popup to ask "Update Property Files", click "Only Once" to dismiss.

10. Double check AndroidManifest.xml is still same.

11. From the menu, click Build > Rebuild Project.