Android Google Map

Android provides facility to integrate Google map in our application. Google map displays your current location, navigate location direction, search location etc. We can also customize Google map according to our requirement.

Types of Google Maps

There are four different types of Google maps, as well as an optional to no map at all. Each of them gives different view on map. These maps are as follow:

1. **Normal:** This type of map displays typical road map, natural features like river and some features build by humans.
2. **Hybrid:** This type of map displays satellite photograph data with typical road maps. It also displays road and feature labels.
3. **Satellite:** Satellite type displays satellite photograph data, but doesn't display road and feature labels.
4. **Terrain:** This type displays photographic data. This includes colors, contour lines and labels and perspective shading.
5. **None:** This type displays an empty grid with no tiles loaded.

Syntax of different types of map

1. googleMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);
2. googleMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);
3. googleMap.setMapType(GoogleMap.MAP\_TYPE\_SATELLITE);
4. googleMap.setMapType(GoogleMap.MAP\_TYPE\_TERRAIN);

Methods of Google map

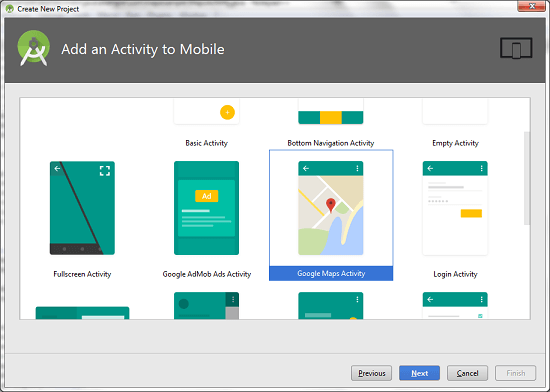
Google map API provides several methods that help to customize Google map. These methods are as following:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| addCircle(CircleOptions options) | This method add circle to map. |
| addPolygon(PolygonOptions options) | This method add polygon to map. |
| addTileOverlay(TileOverlayOptions options) | This method add tile overlay to the map. |
| animateCamera(CameraUpdate update) | This method moves the map according to the update with an animation. |
| clear() | This method removes everything from the map. |
| getMyLocation() | This method returns the currently displayed user location. |
| moveCamera(CameraUpdate update) | This method reposition the camera according to the instructions defined in the update. |
| setTrafficEnabled(boolean enabled) | This method set the traffic layer on or off. |
| snapshot(GoogleMap.SnapshotReadyCallback callback) | This method takes a snapshot of the map. |
| stopAnimation() | This method stops the camera animation if there is any progress. |

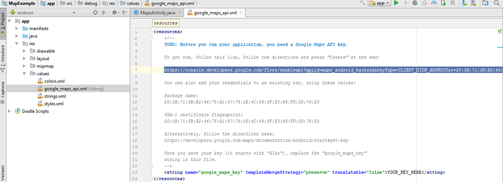
Example of Google Map

Let's create an example of Google map integrating within our app. For doing this we select Google Maps Activity.

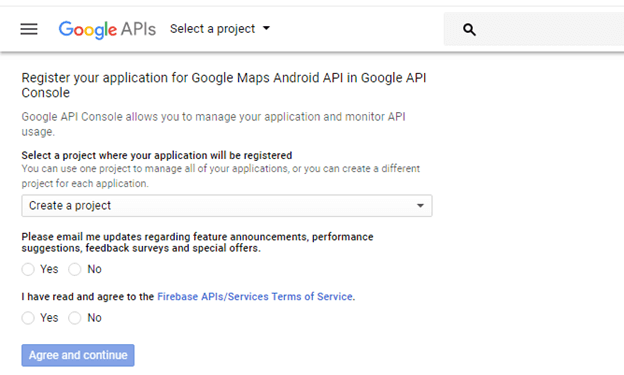
Play Video[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)



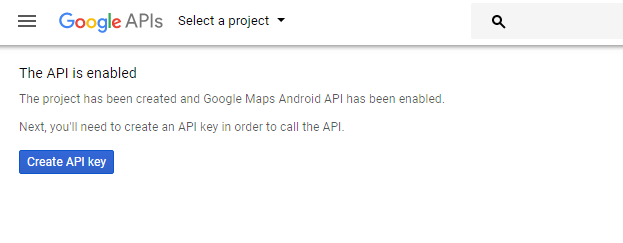
Copy the URL from google\_map\_api.xml file to generate Google map key.



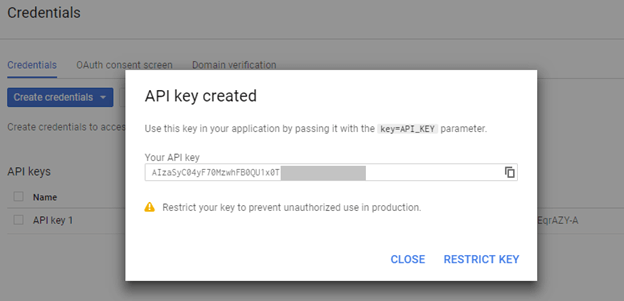
Paste the copied URL at the browser. It will open the following page.



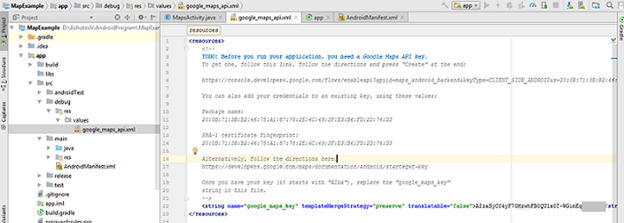
Click on Create API key to generate API key.



After clicking on Create API key, it will generate our API key displaying the following screen.



Copy this generated API key in our *google\_map\_api.xml* file



activity\_maps.xml

1. **<fragment** xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:map="http://schemas.android.com/apk/res-auto"
3. xmlns:tools="http://schemas.android.com/tools"
4. android:id="@+id/map"
5. android:name="com.google.android.gms.maps.SupportMapFragment"
6. android:layout\_width="match\_parent"
7. android:layout\_height="match\_parent"
8. tools:context="example.com.mapexample.MapsActivity" **/>**

MapsActivity.java

To get the GoogleMap object in our MapsActivity.java class we need to implement the OnMapReadyCallback interface and override the onMapReady() callback method.

1. **package** example.com.mapexample;
2. **import** android.support.v4.app.FragmentActivity;
3. **import** android.os.Bundle;
4. **import** com.google.android.gms.maps.CameraUpdateFactory;
5. **import** com.google.android.gms.maps.GoogleMap;
6. **import** com.google.android.gms.maps.OnMapReadyCallback;
7. **import** com.google.android.gms.maps.SupportMapFragment;
8. **import** com.google.android.gms.maps.model.LatLng;
9. **import** com.google.android.gms.maps.model.MarkerOptions;
10. **public** **class** MapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback{
11. **private** GoogleMap mMap;
12. @Override
13. **protected** **void** onCreate(Bundle savedInstanceState) {
14. **super**.onCreate(savedInstanceState);
15. setContentView(R.layout.activity\_maps);
16. // Obtain the SupportMapFragment and get notified when the map is ready to be used.
17. SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
18. .findFragmentById(R.id.map);
19. mapFragment.getMapAsync(**this**);
20. }
21. @Override
22. **public** **void** onMapReady(GoogleMap googleMap) {
23. mMap = googleMap;
24. // Add a marker in Sydney and move the camera
25. LatLng sydney = **new** LatLng(-34, 151);
26. mMap.addMarker(**new** MarkerOptions().position(sydney).title("Marker in Sydney"));
27. mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
28. }
29. }

Required Permission

Add the following user-permission in AndroidManifest.xml file.

1. **<uses-permission** android:name="android.permission.ACCESS\_FINE\_LOCATION" **/>**
2. **<uses-permission** android:name="android.permission.ACCESS\_COARSE\_LOCATION" **/>**
3. **<uses-permission** android:name="android.permission.INTERNET" **/>**

AndroidManifest.xml

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
3. package="example.com.mapexample"**>**
4. <!--
5. The ACCESS\_COARSE/FINE\_LOCATION permissions are not required to use
6. Google Maps Android API v2, but you must specify either coarse or fine
7. location permissions for the 'MyLocation' functionality.
8. --**>**
9. **<uses-permission** android:name="android.permission.ACCESS\_FINE\_LOCATION" **/>**
10. **<uses-permission** android:name="android.permission.ACCESS\_COARSE\_LOCATION" **/>**
11. **<uses-permission** android:name="android.permission.INTERNET" **/>**
12. **<application**
13. android:allowBackup="true"
14. android:icon="@mipmap/ic\_launcher"
15. android:label="@string/app\_name"
16. android:roundIcon="@mipmap/ic\_launcher\_round"
17. android:supportsRtl="true"
18. android:theme="@style/AppTheme"**>**
19. **<meta-data**
20. android:name="com.google.android.geo.API\_KEY"
21. android:value="@string/google\_maps\_key" **/>**
22. **<activity**
23. android:name=".MapsActivity"
24. android:label="@string/title\_activity\_maps"**>**
25. **<intent-filter>**
26. **<action** android:name="android.intent.action.MAIN" **/>**
27. **<category** android:name="android.intent.category.LAUNCHER" **/>**
28. **</intent-filter>**
29. **</activity>**
30. **</application>**
31. **</manifest>**

build.gradel

Add the following dependencies in *build.gradel* file.

1. dependencies {
2. implementation fileTree(dir: 'libs', include: ['\*.jar'])
3. implementation 'com.android.support:appcompat-v7:26.1.0'
4. implementation 'com.google.android.gms:play-services-maps:11.8.0'
5. testImplementation 'junit:junit:4.12'
6. androidTestImplementation 'com.android.support.test:runner:1.0.1'
7. androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
8. }

Output

