Android Google Map Displaying Current Location

In the previous tutorial of Android Google Map, we simply displayed the default coordinates (location) set by the *MapsActivity.java* class file.

Now in this tutorial we will display and place marker at the user current location. For doing this we need to generate Google Map API key. The process of generating Google Map API is described in tutorial Android Google Map.

To display the user current location we need to implements some interfaces and there callbacks methods.

Callback methods in Google Map

1. **OnMapRreadyCallback:** This callback interface invokes when it instance is set on MapFragment object. The onMapReady(GoogleMap) method of OnMapReadyCallback interface is called when the map is ready to used. In the onMapReady(GoogleMap) method we can add markers, listeners and other attributes.
2. **LocationListener:** This interface is used to receive notification when the device location has changed. The abstract method of LocationListener onLocationChanged(Location) is called when the location has changed.
3. **GoogleApiClient.ConnectionCallbacks:** This interface provide callbacks methods onConnected(Bundle) and onConnectionSuspended(int) which are called when the device is to connected and disconnected.
4. **GoogleApiClient.OnConnectionFailedListener:** This interface provide callbacks method onConnectionFailed(ConnectionResult) which is called when there was an error in connecting the device to the service.

The **setMyLocationEnabled()** method of GoogleMap is used to enable location layer, which allows device to interact with current location.

Example of Google Map Displaying Current Location

Let's see an example of Google Map which displays the current location of device.

activity\_maps.xml

Add a SupportMapFragment in fragment in activity\_maps.xml file.

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" **/>**

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. compile 'com.google.android.gms:play-services-location:11.8.0'
6. testImplementation 'junit:junit:4.12'
7. androidTestImplementation 'com.android.support.test:runner:1.0.1'
8. androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
10. }

MapsActivity.java

Add the following code in MapsActivity.java file.

1. **package** example.com.mapexample;

4. **import** android.os.Build;
5. **import** android.support.v4.app.FragmentActivity;
6. **import** android.os.Bundle;
8. **import** com.google.android.gms.common.api.GoogleApiClient;
9. **import** com.google.android.gms.maps.CameraUpdateFactory;
10. **import** com.google.android.gms.maps.GoogleMap;
11. **import** com.google.android.gms.maps.OnMapReadyCallback;
12. **import** com.google.android.gms.maps.SupportMapFragment;
13. **import** com.google.android.gms.maps.model.BitmapDescriptorFactory;
14. **import** com.google.android.gms.maps.model.LatLng;
15. **import** com.google.android.gms.maps.model.Marker;
16. **import** com.google.android.gms.maps.model.MarkerOptions;
17. **import** com.google.android.gms.location.LocationServices;
19. **import** android.location.Location;
20. **import** android.Manifest;
21. **import** android.content.pm.PackageManager;
22. **import** android.support.v4.content.ContextCompat;
23. **import** com.google.android.gms.common.ConnectionResult;
24. **import** com.google.android.gms.location.LocationListener;
25. **import** com.google.android.gms.location.LocationRequest;

28. **public** **class** MapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback,
29. LocationListener,GoogleApiClient.ConnectionCallbacks,
30. GoogleApiClient.OnConnectionFailedListener{
32. **private** GoogleMap mMap;
33. Location mLastLocation;
34. Marker mCurrLocationMarker;
35. GoogleApiClient mGoogleApiClient;
36. LocationRequest mLocationRequest;
38. @Override
39. **protected** **void** onCreate(Bundle savedInstanceState) {
40. **super**.onCreate(savedInstanceState);
41. setContentView(R.layout.activity\_maps);
42. // Obtain the SupportMapFragment and get notified when the map is ready to be used.
43. SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
44. .findFragmentById(R.id.map);
45. mapFragment.getMapAsync(**this**);
47. }
49. @Override
50. **public** **void** onMapReady(GoogleMap googleMap) {
51. mMap = googleMap;
53. **if** (android.os.Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {
54. **if** (ContextCompat.checkSelfPermission(**this**,
55. Manifest.permission.ACCESS\_FINE\_LOCATION)
56. == PackageManager.PERMISSION\_GRANTED) {
57. buildGoogleApiClient();
58. mMap.setMyLocationEnabled(**true**);
59. }
60. }
61. **else** {
62. buildGoogleApiClient();
63. mMap.setMyLocationEnabled(**true**);
64. }
66. }
67. **protected** **synchronized** **void** buildGoogleApiClient() {
68. mGoogleApiClient = **new** GoogleApiClient.Builder(**this**)
69. .addConnectionCallbacks(**this**)
70. .addOnConnectionFailedListener(**this**)
71. .addApi(LocationServices.API).build();
72. mGoogleApiClient.connect();
73. }
75. @Override
76. **public** **void** onConnected(Bundle bundle) {
78. mLocationRequest = **new** LocationRequest();
79. mLocationRequest.setInterval(1000);
80. mLocationRequest.setFastestInterval(1000);
81. mLocationRequest.setPriority(LocationRequest.PRIORITY\_BALANCED\_POWER\_ACCURACY);
82. **if** (ContextCompat.checkSelfPermission(**this**,
83. Manifest.permission.ACCESS\_FINE\_LOCATION)
84. == PackageManager.PERMISSION\_GRANTED) {
85. LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, **this**);
86. }
88. }
90. @Override
91. **public** **void** onConnectionSuspended(**int** i) {
93. }
95. @Override
96. **public** **void** onLocationChanged(Location location) {
98. mLastLocation = location;
99. **if** (mCurrLocationMarker != **null**) {
100. mCurrLocationMarker.remove();
101. }
102. //Place current location marker
103. LatLng latLng = **new** LatLng(location.getLatitude(), location.getLongitude());
104. MarkerOptions markerOptions = **new** MarkerOptions();
105. markerOptions.position(latLng);
106. markerOptions.title("Current Position");
107. markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_GREEN));
108. mCurrLocationMarker = mMap.addMarker(markerOptions);
110. //move map camera
111. mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
112. mMap.animateCamera(CameraUpdateFactory.zoomTo(11));
114. //stop location updates
115. **if** (mGoogleApiClient != **null**) {
116. LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, **this**);
117. }
119. }
121. @Override
122. **public** **void** onConnectionFailed(ConnectionResult connectionResult) {
124. }
126. }

Request Runtime Permission

Android device having Android 6.0 (Marshmallow) or later are required some permission at runtime to access device functionality.

In the above MapsActivity.java file we added a runtime permission *Manifest.permission.ACCESS\_FINE\_LOCATION* which request to access device location. The runtime permission is checked using *checkSelfPermission()* method and return *PackageManager.PERMISSION\_GRANTED* or *PackageManager.PERMISSION\_DENIED*. If permission granted than app proceeds for operation.

Required Permission in AndroidManifest.xml

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" **/>**

14. **<application**
15. android:allowBackup="true"
16. android:icon="@mipmap/ic\_launcher"
17. android:label="@string/app\_name"
18. android:roundIcon="@mipmap/ic\_launcher\_round"
19. android:supportsRtl="true"
20. android:theme="@style/AppTheme"**>**
21. <!--
22. The API key for Google Maps-based APIs is defined as a string resource.
23. (See the file "res/values/google\_maps\_api.xml").
24. Note that the API key is linked to the encryption key used to sign the APK.
25. You need a different API key for each encryption key, including the release key that is used to
26. sign the APK for publishing.
27. You can define the keys for the debug and release targets in src/debug/ and src/release/.
28. --**>**
29. **<meta-data**
30. android:name="com.google.android.geo.API\_KEY"
31. android:value="@string/google\_maps\_key" **/>**
33. **<activity**
34. android:name=".MapsActivity"
35. android:label="@string/title\_activity\_maps"**>**
36. **<intent-filter>**
37. **<action** android:name="android.intent.action.MAIN" **/>**
39. **<category** android:name="android.intent.category.LAUNCHER" **/>**
40. **</intent-filter>**
41. **</activity>**
42. **</application>**
44. **</manifest>**

Output