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

```
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

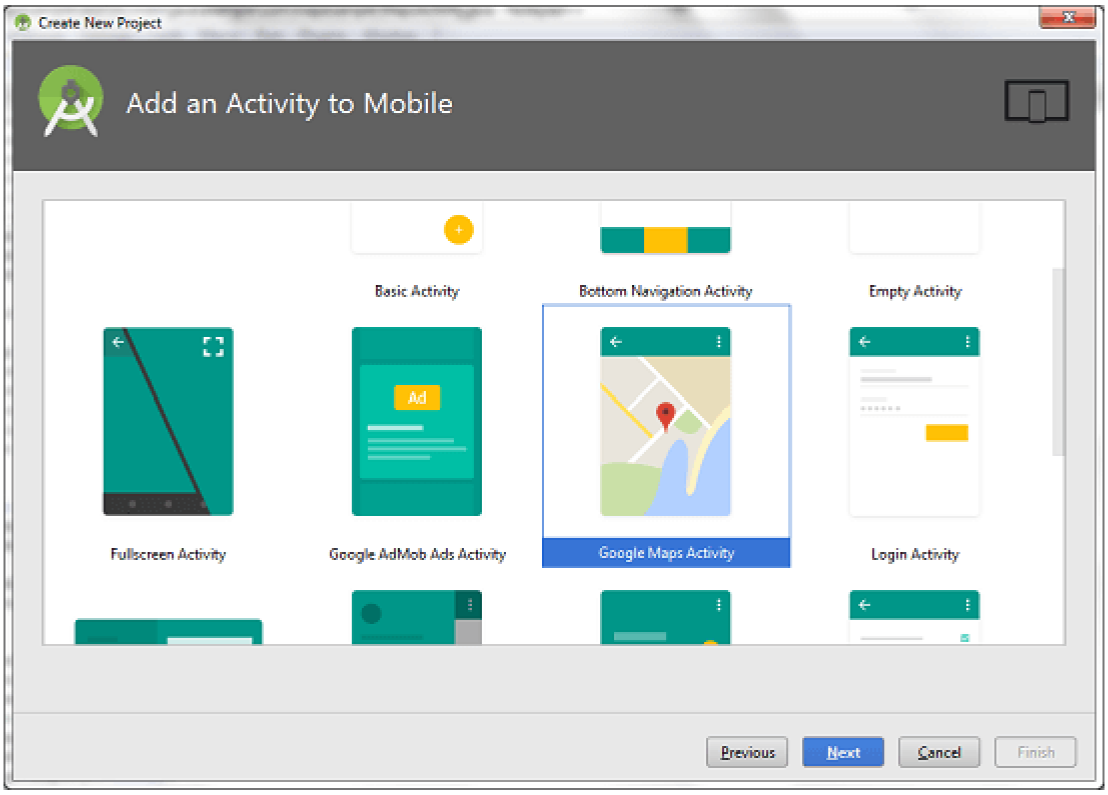
Methods of Google map

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

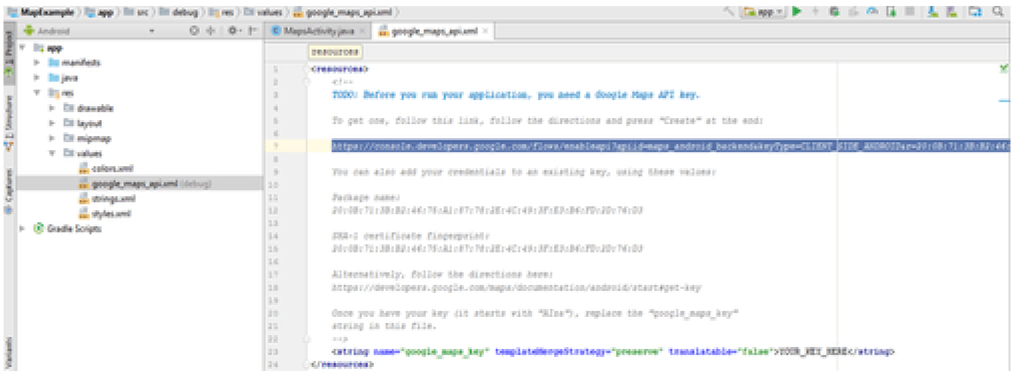
Methods	Description
addCircle(CircleOptions options)	This method add circle to map.
addPolygon(PolygonOptions options)	This method add polygon to map.
addTileOverlay(TileOverlayOptions options)	This method add tile overlay to the map.
animateCamera(CameraUpdate update)	This method moves the map according 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.

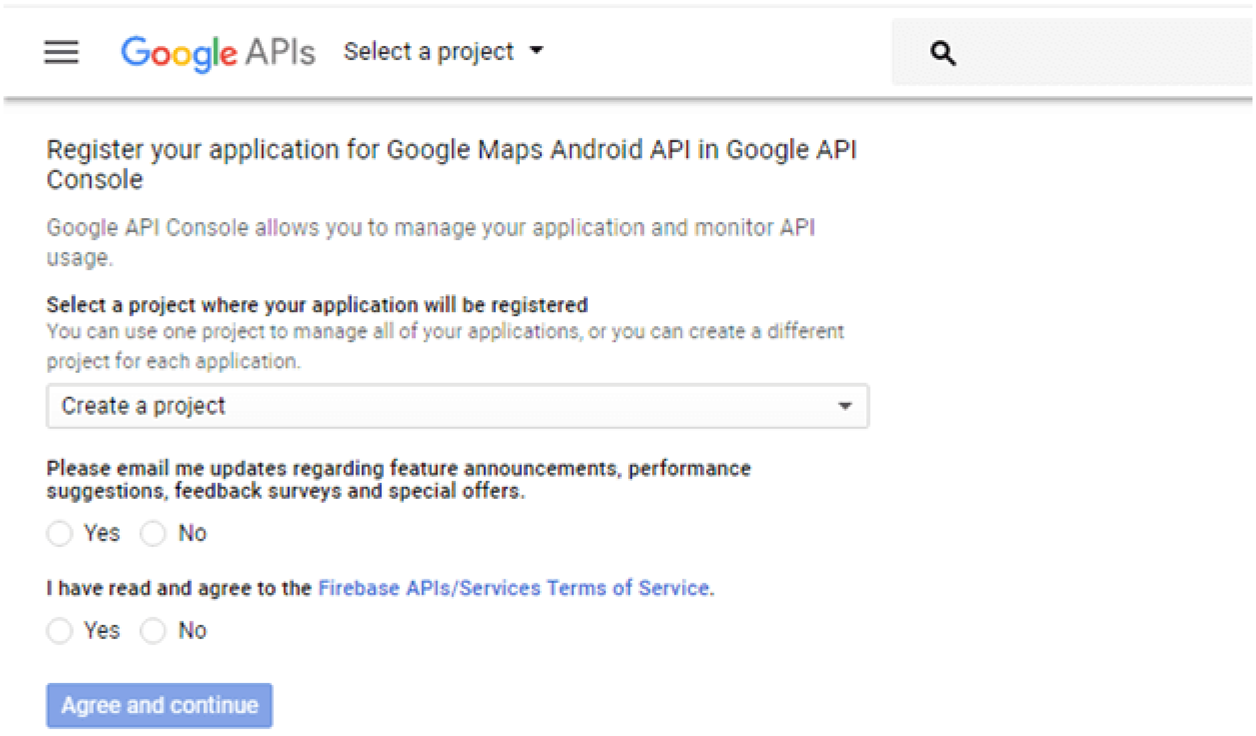


Copy the URL from google_map_api.xml file to generate Google map key.



```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=[illegible]

    You can also add your credentials to an existing key, using these values:

    Package name:
    [illegible]

    SHA-1 certificate fingerprint:
    [illegible]

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>
```

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


Register your application for Google Maps Android API in Google API Console

Google API Console allows you to manage your application and monitor API usage.

Select a project where your application will be registered
You can use one project to manage all of your applications, or you can create a different project for each application.

Create a project

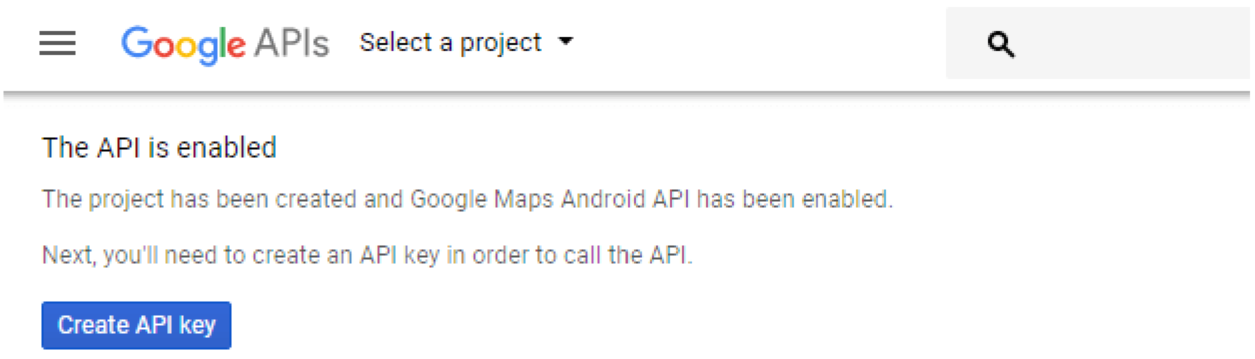
Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

Yes No

I have read and agree to the Firebase APIs/Services Terms of Service.

Yes No

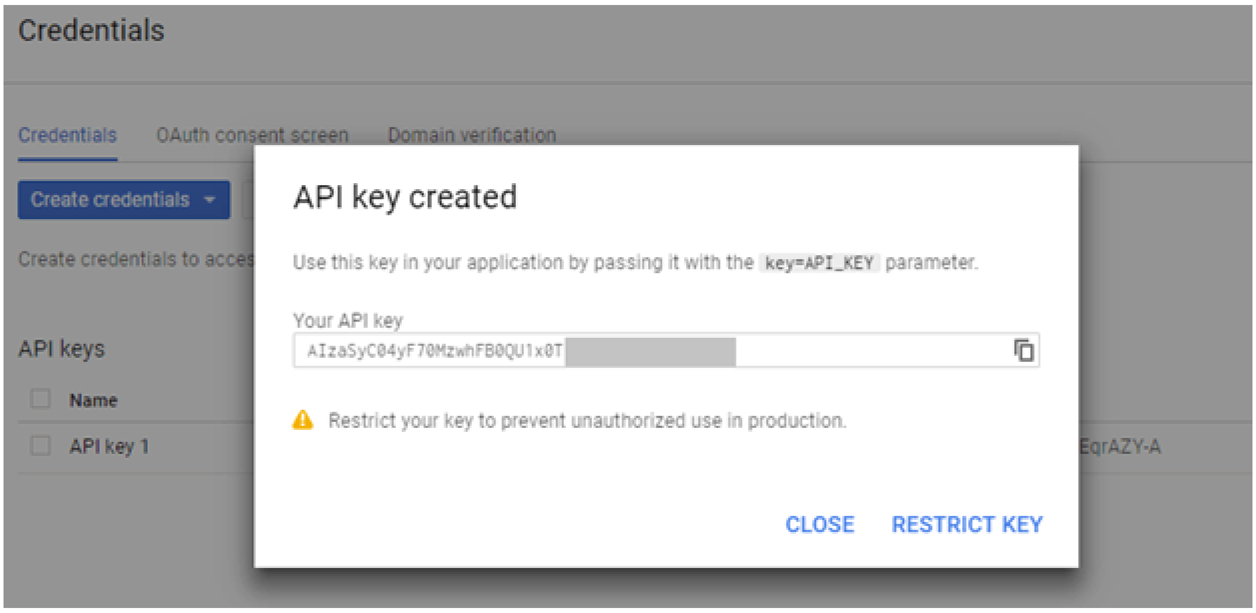
Click on Create API key to generate API key.


The API is enabled

The project has been created and Google Maps Android API has been enabled.

Next, you'll need to create an API key in order to call the API.

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



Credentials

Credentials   OAuth consent screen   Domain verification

Create credentials

Create credentials to acces

API keys

Name

API key 1

EqrAZY-A

API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

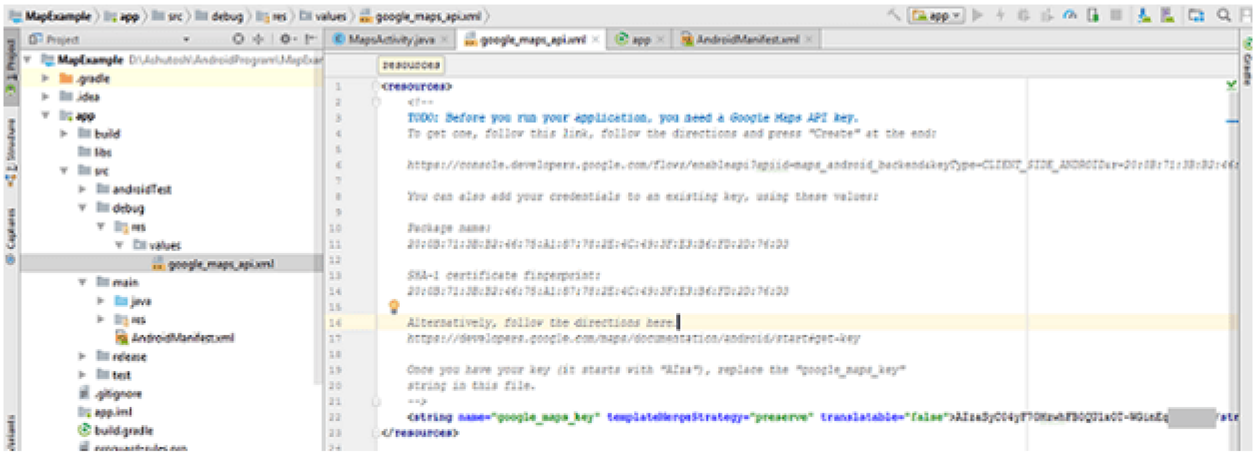
Your API key

AIzaSyC04yF70MzwhFB0QU1x0T[redacted]

Restrict your key to prevent unauthorized use in production.

CLOSE   RESTRICT KEY

Copy this generated API key in our google_map_api.xml file



```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.
    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=20:0B:71:3B:B2:46

    You can also add your credentials to an existing key, using these values:

    Package name:
    20:0B:71:3B:B2:46:75:A1:67:78:2E:4C:49:3F:E3:B6:FD:2D:76:D3

    SHA-1 certificate fingerprint:
    20:0B:71:3B:B2:46:75:A1:67:78:2E:4C:49:3F:E3:B6:FD:2D:76:D3

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyC04yF70MzwhFB0QU1x0T-WGiuEq[redacted]</str
</resources>
```

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.

```
package example.com.mapexample;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback{
    private GoogleMap mMap;
      @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);          // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
```

```
            // Add a marker in Sydney and move the camera
            LatLng sydney = new LatLng(-34, 151);
mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
            mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));

        }
    }
```

Required Permission

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

```
    <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" />
```

AndroidManifest.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="example.com.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" />
```

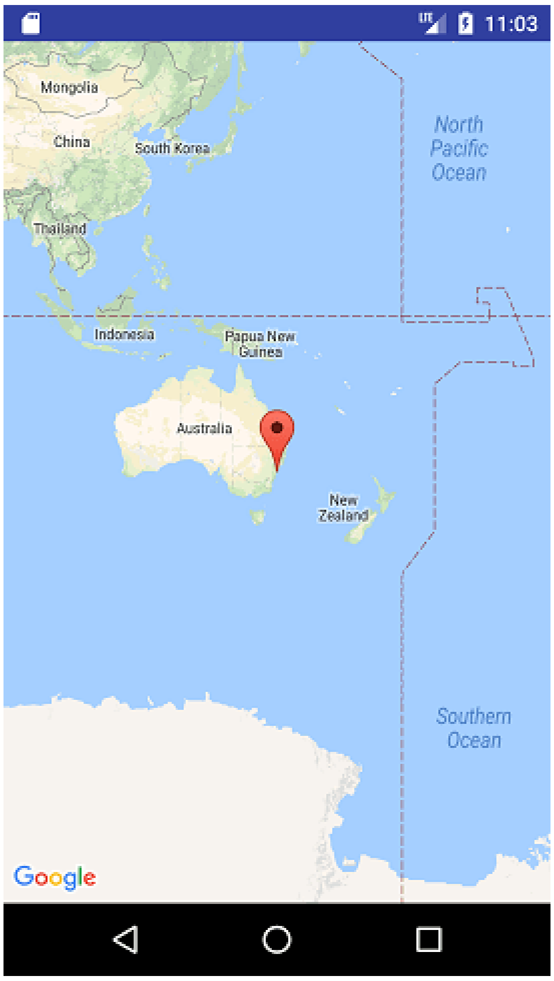
```
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_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>
```

build.gradel

Add the following dependencies in build.gradel file.

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

Output


11:03
Mongolia
China
South Korea
Thailand
Indonesia
Papua New Guinea
Australia
New Zealand
North Pacific Ocean
Southern Ocean
Google