Android Maps Tutorial

Introduction: Using Google Maps API

In this project, we are going to create a project that will show a user inputted address on a Google map. The main Activity will contain an area for the map, an EditText component for entering the address and a button that is clicked to initiate the displaying of the updated map. Figure 1 shows the final project running on an Android Virtual Device.

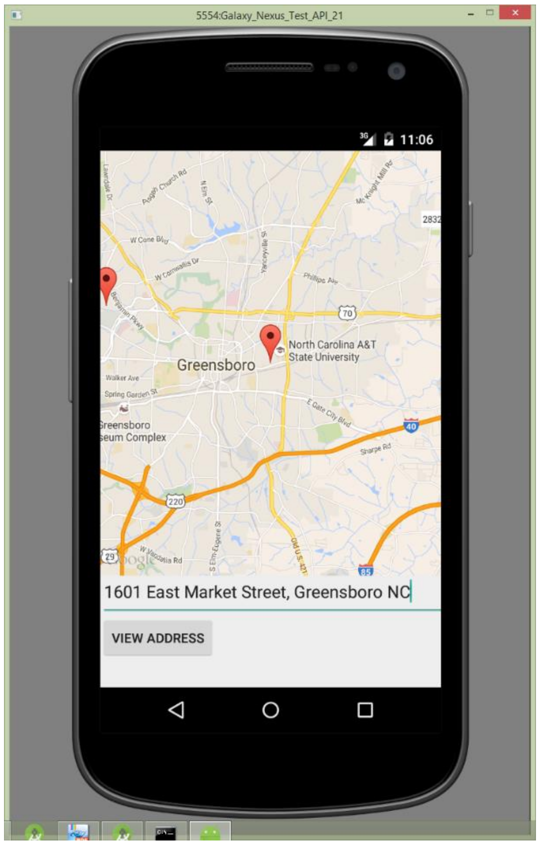

5554:Galaxy_Nexus_Test_API_21
11:06
Greensboro
North Carolina A&T State University
1601 East Market Street, Greensboro NC
VIEW ADDRESS

Figure 1 – Final App screenshot

Prerequisites:

- It is assumed that you have already completed the introductory lab and are familiar with creating projects in Android Studio.
- Make sure that all the Google Play items are installed under the "Extras" section in the Android SDK Manager.
- You will need a Google account to obtain a Maps API key to allow you to access Google Maps.
- If you will be using a virtual device, create one with the following characteristics:
    - **Category:** Phone
    - **Phone Name**: Galaxy Nexus

- **System Image**: Marshmellow (API Level 23), x86, **Google APIs** (This is important – Do not use the Android platforms)
- **AVD Name:** Input whatever you like (e.g. Galaxy Nexus Test Phone).

Phase 1: Simple Map

The first phase of this project is to create an app that shows a map centered at location (Lat: 0, Long: 0). This will allow us to see if we have selected the right API level, have our API Key setup correctly and manifest file setup with all the correct permissions.

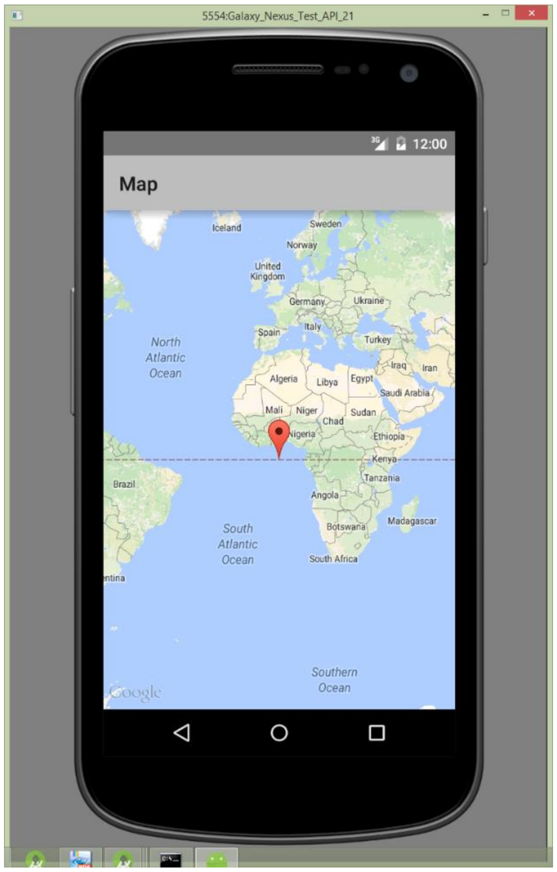


Figure 2 – Simple Google Map App

Step 1:

Create a new project and name it something like MapsApp. Other than the Application name, keep everything else at the default values. Click next, then select the “Phone and Tablet” check box. The Minimum SDK level should be API: 23 (Marshmellow). Click next and then select the Google Maps Activity. Click next and use the defaults on the next page. Click Finish.

Step 2:

Follow the instructions in the xml document that appears to obtain your Maps API key and insert it at the indicated location.

Step 3:

Now run the app on the AVD that was mentioned in the Prerequisite section. A nice map should appear!

Phase 2: Move Marker and Camera

Now, we will the ability to move the map marker to a new location and update the camera so that the new location is a the center of the device screen.

Step 1:

Create a new instance variable (newLocation) of type LatLng in the MapsActivity class. Note, if you put the cursor on LatLng the IDE will add the import for you. Instantiate the variable in the onCreate method using a known LatLng location. You can use google to find the (lat, long) of where ever you would like your app to pull up. For example, I entered my home address followed by the words "longitude latitude" in a google search.

Step 2:

Modify the setUpMap method to move the marker to newLocation and update the camera to newLocation:

```
CameraUpdate yourLocation = CameraUpdateFactory.newLatLngZoom(newLocation, 12);
mMap.animateCamera(yourLocation);
mMap.addMarker(new MarkerOptions().position(newLocation).title("Marker"));
```

The above code also animates the camera zoom to level 12.

Step 3:

Run the updated app to make sure that your requested location is displayed.

Phase 3: Final App

In this phase, we will update the main Activity to also display an EditText object and button. We will add the code that will handle the button click event, converting the given address into LatLng coordinates and updating the map to show the given location centered on the screen.

Step 1:

Update the activity_maps.xml file so that it has a LinearLayout containing the maps fragment as a sub-element. You need to click on the "Text" tab and then enclose the current file contents within the layout tag:

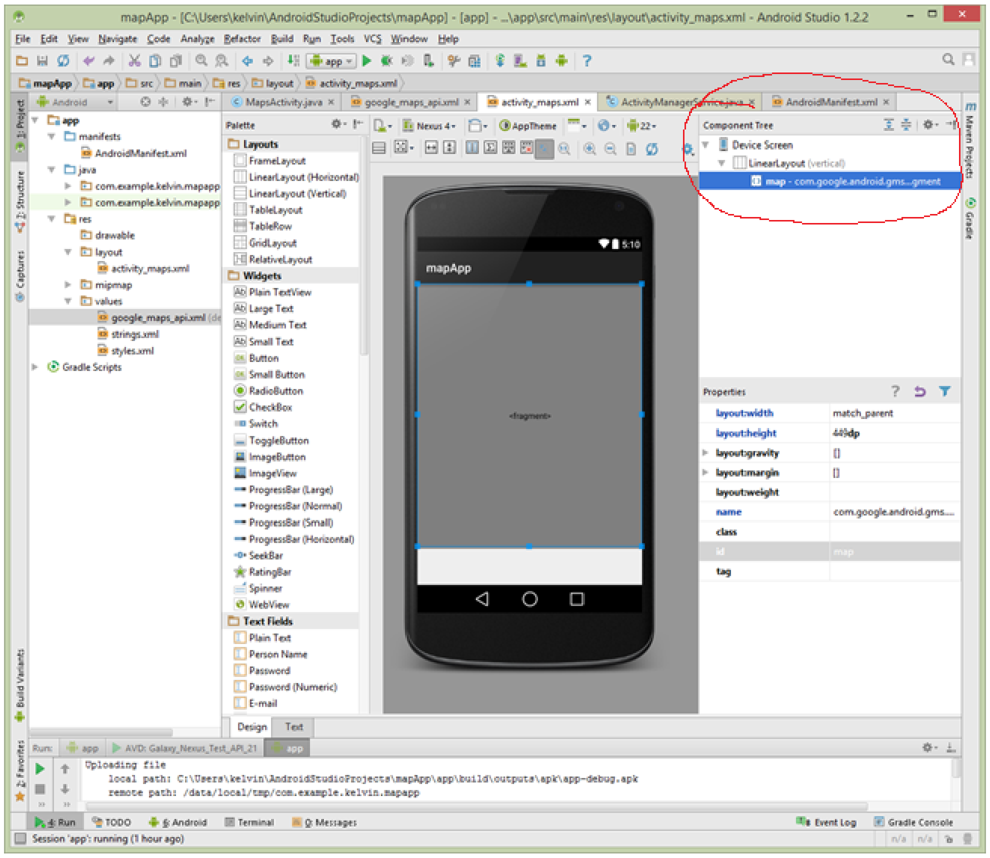
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
```

Replace with current file contents containing fragment declaration

</LinearLayout>

Step 2:

Resize the map fragment so that you will have room to add the EditText and Button elements. Now drag and drop the EditText and Button objects uder the map fragment in the Component Tree (far right window – See Figure 3). Give the objects appropriate names.



```
Uploading file
    local path: C:\Users\kelvin\AndroidStudioProjects\mapApp\app\build\outputs\apk\app-debug.apk
    remote path: /data/local/tmp/com.example.kelvin.mapapp
```

layout:width	match_parent
layout:height	449dp
layout:gravity	[]
layout:margin	[]
layout:weight	
name	com.google.android.gms...
class	
id	map
tag	

Figure 3 – Android Studio Component Tree Location

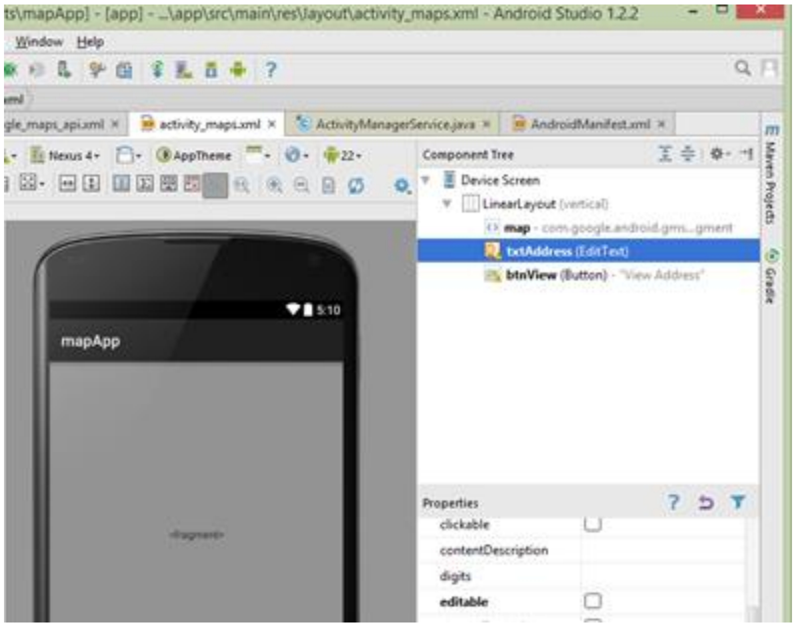


Figure 4 – Updated Component Tree

Step 3:

Go to the onCreate method in the MapsActivity class and add code to create a reference to the button object added to the Activity. Create an anonymous OnClickListener that creates a reference to the EditText, converts the address in the EditText to a LatLng object, assigns the LatLng to the newLocation variable and class setUpMap to update the map. Hint: You must perform a toString() on the return value of the getText() method for the EditText class. Here is the method that converts the string address to a LatLng Object.

```
public LatLng getLocationFromAddress(String strAddress) {

    Geocoder coder = new Geocoder(this);
    List<Address> address;
    LatLng p1 = null;

    try {
        address = coder.getFromLocationName(strAddress, 5);
        if (address == null) {
            return null;
        }
        Address location = address.get(0);
        location.getLatitude();
        location.getLongitude();

        p1 = new LatLng(location.getLatitude(), location.getLongitude() );

    } catch (Exception ex) {

        ex.printStackTrace();
    }

    return p1;
}
```

*** If your app has problems executing, restart the AVD.

Step 4:

Run the final application. Have fun!!!

References:

1. http://blog.teamtreehouse.com/beginners-guide-location-android
2. http://www.joellipman.com/articles/google/android-o-s/app-development/733-basic-android-app-using-google-maps-and-current-location.html
3. http://www.codota.com/android/classes/android.location.Geocoder
4. https://developer.android.com/tools/debugging/debugging-studio.html
5. http://developer.android.com/reference/android/location/Geocoder.html
6. http://developer.android.com/reference/android/location/LocationListener.html#onLocationChanged%28android.location.Location%29
7. https://developers.google.com/maps/documentation/android/views
8. http://javapapers.com/android/android-location-tracker-with-google-maps/
9. https://developers.google.com/maps/documentation/android/marker#add_a_marker
10. http://www.androidhive.info/2012/07/android-gps-location-manager-tutorial/
11. http://stackoverflow.com/questions/3574644/how-can-i-find-the-latitude-and-longitude-from-address