

Programming in Android



Session: 9

Google API

Objectives

- ◆ Explain API and its uses
- ◆ Explain Google API
- ◆ Explain location based service



Introduction

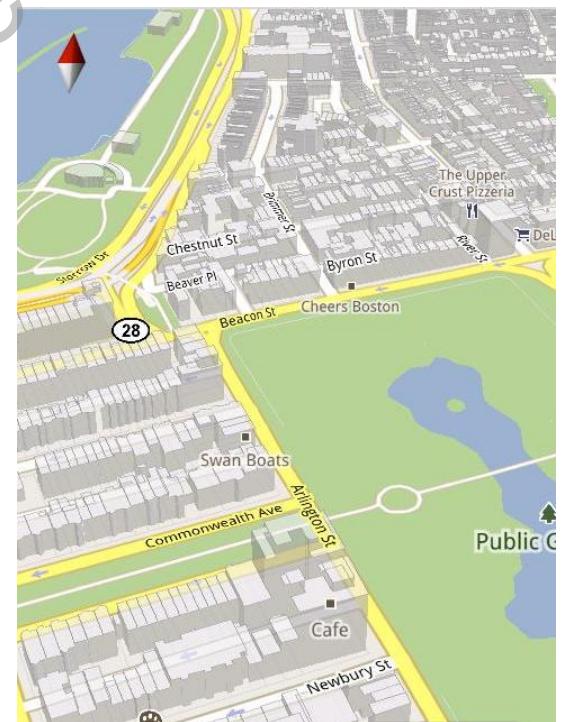
- ◆ An API can be explained as follows:
 - ❖ API stands for Application Programming Interface
 - ❖ It is a specification used by software components to communicate with each other
 - ❖ It is a library containing specifications for objects, classes, variables, and so on
 - ❖ It describes the ways in which a particular task is performed
- ◆ Google API's basically consist of specialized Web services, programs, and specialized scripts
- ◆ Google APIs are used as an added resource in their applications

- ◆ Google as an organization provides several services such as Google Drive, Google App Engine, Google Translate, Google Maps, and so on
- ◆ Developers can integrate these services provided by Google into their application free of cost
- ◆ The developer needs to keep in mind that by using Google API, the application will rely on a network connection in order to function properly



Working with Location Based Services

- ◆ Location Based Service (LBS) is an information service and has a number of uses in social networking
- ◆ With the incorporation of Global Positioning System (GPS) devices in Smartphones, LBS have become important in the past few years
- ◆ Most important classes and interface that are to be used are as follows:
 - ❖ LocationManager
 - ❖ LocationProvider
 - ❖ Location
 - ❖ LocationListener



Retrieving Current Location Example

- ◆ The user analytics on the map are feasible in one of the following ways:
 - ◆ Using the GPS device that comes with the mobile
 - ◆ Using the ID of the cell that the user is currently being served by
- ◆ The current user location can be retrieved as shown in the following Code Snippet:

```
LocationManager locationManager =  
(LocationManager) getSystemService(Context.LOCATION_SERVICE);  
  
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  
MINIMUM_TIME_BETWEEN_UPDATES, MINIMUM_DISTANCE_CHANGE_FOR_UPDATES, new  
MyLocationListener());  
  
Location location = locationManager.getLastKnownLocation(LocationManager.GPS_  
PROVIDER);
```

Location Based Services Example

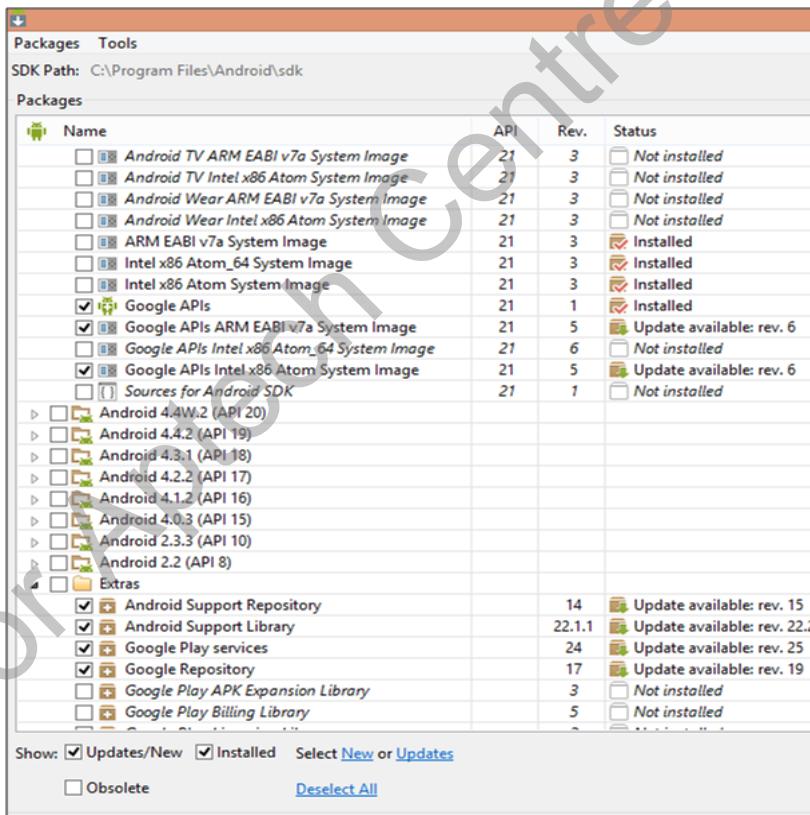
- Using the code, an application for demonstrating LBS is created as shown in the following figure:



- Clicking Retrieve Location will return the current location

Setting up Android SDK for Google API

- ◆ Navigate to Tools → Android and select SDK Manager
- ◆ Select Google APIs under the Android API being used. In this case it is Android 5.0.1
- ◆ Select Google Play services under Extras
- ◆ Click Install. The selected packages will be installed as shown in the given figure

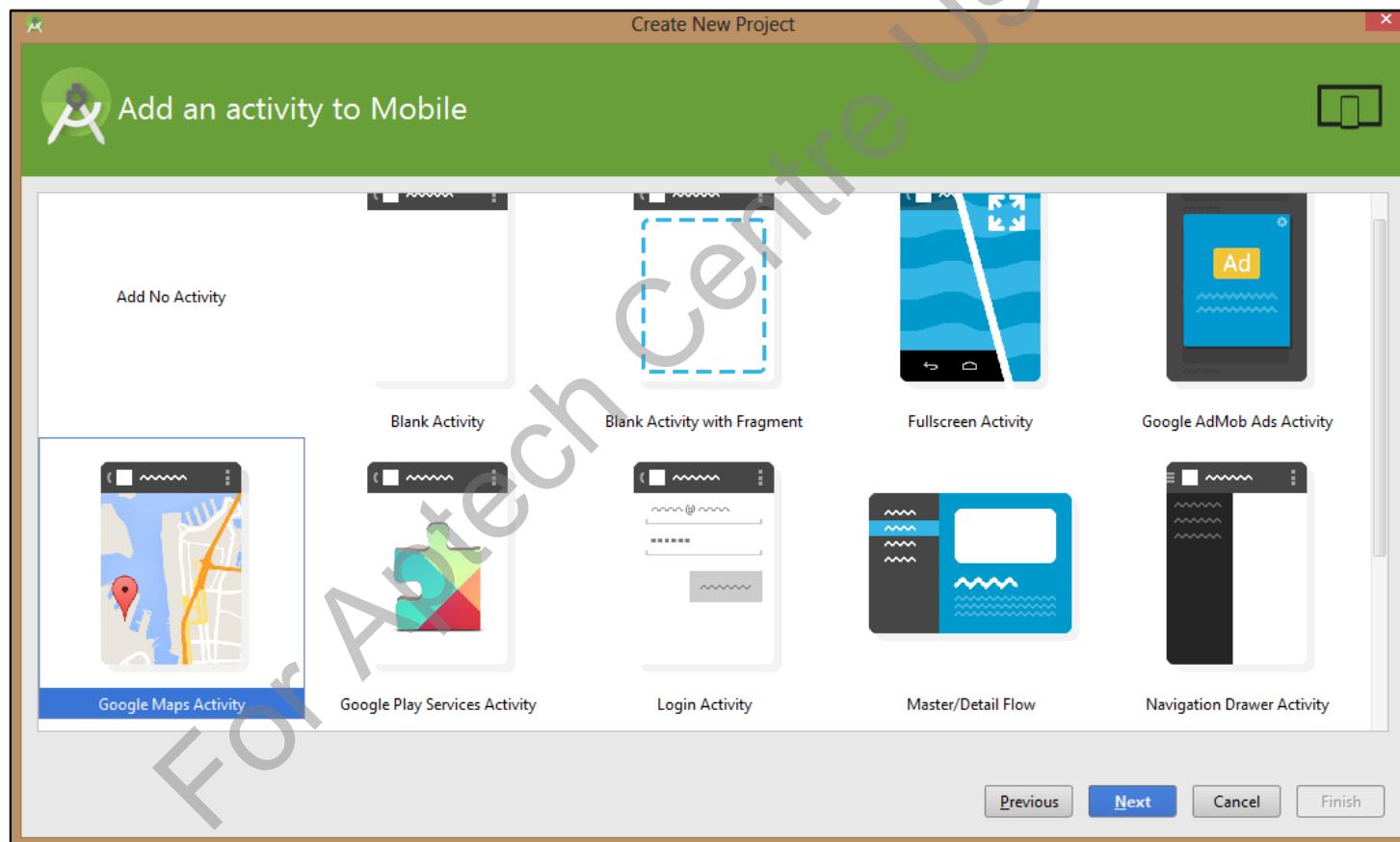


Working with Google Maps

- ◆ Google Maps allows the users to explore the world with rich maps provided by Google
- ◆ It helps to identify locations with custom markers, augment the map data with image overlays, embed one or more maps as fragments, and much more
- ◆ In order to use Google Maps the needs to host an Activity Fragment
- ◆ The Steps are as follows:
 - ◆ The Android SDK should have libraries for Google Maps installed
 - ◆ The Activity class that will be responsible for showing the map needs to extend from ActivityFragment class
 - ◆ The application manifest file needs to be setup with the ‘android.permission.INTERNET’ permission
 - ◆ The application manifest file is also required to be setup with the Maps key

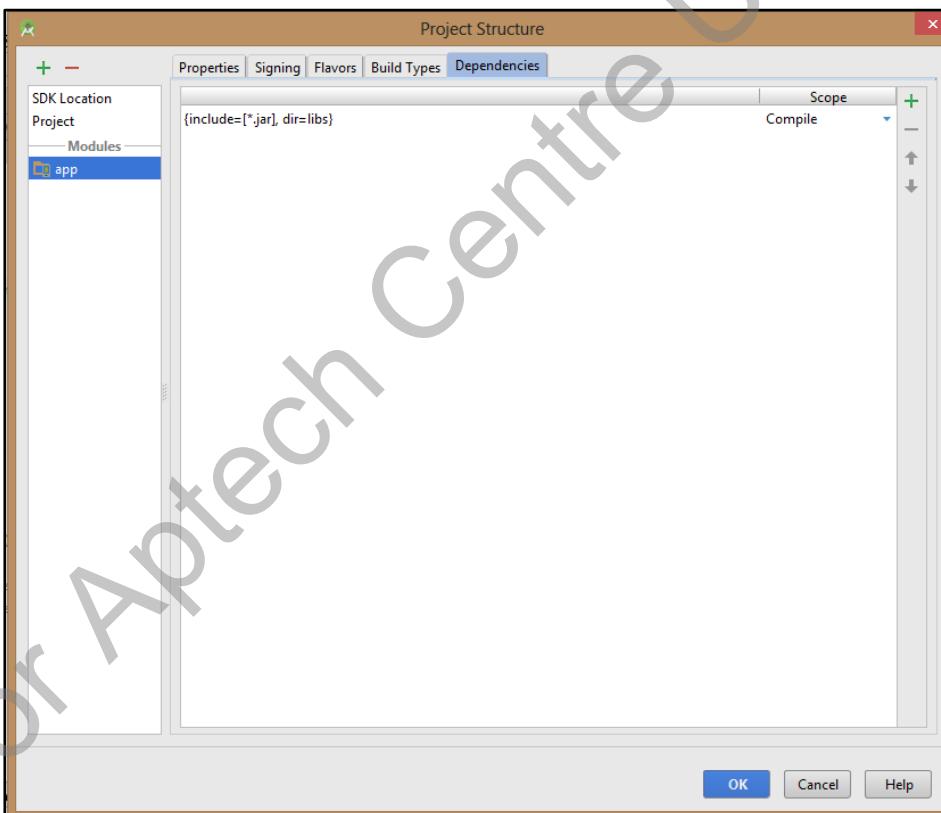
Adding Dependencies for the Application 1-3

- Create a project named GoogleProject in Android Studio
- During the activity selection screen, select Google Maps Activity as shown in the following figure:



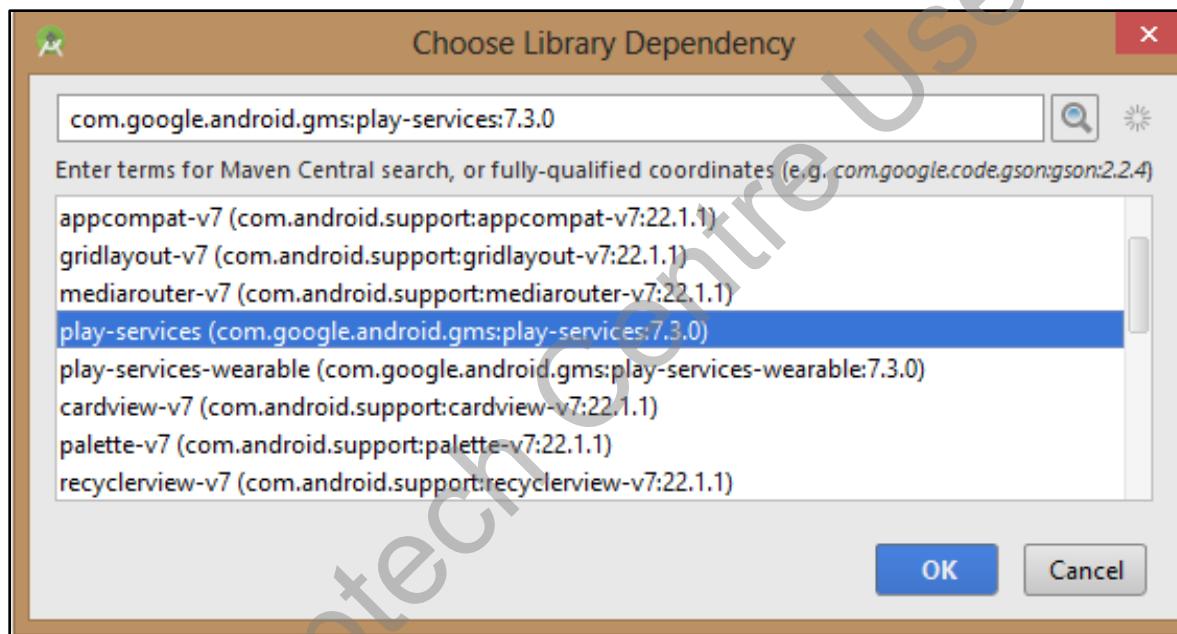
Adding Dependencies for the Application 2-3

- If the project is created using a Blank Activity, right-click app and select Open Module Settings
- From the Project Structure dialog box, select the Dependencies tab as shown in the following figure:



Adding Dependencies for the Application 3-3

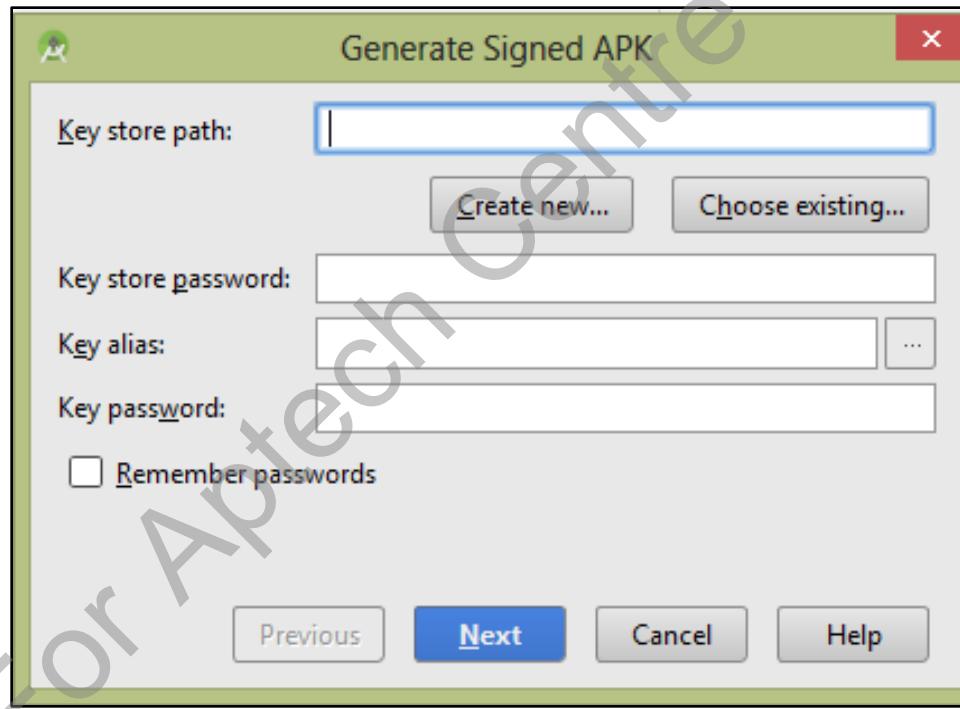
- Click the '+' sign and select Library Dependency to open Library Selection dialog box as shown in the following figure:



- Select the play-services library and click OK
- Click OK again to confirm the dependencies
- Navigate to app → java → com.example.googleproject and add the import for the package com.google.android.gms.maps.GoogleMap

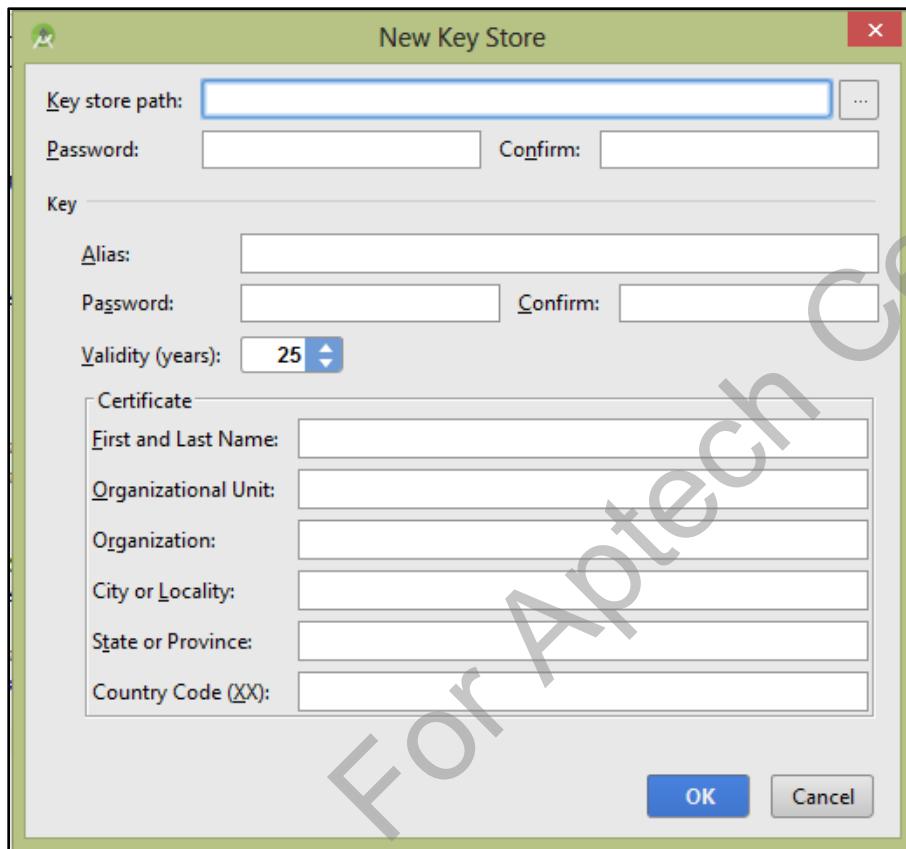
Generating a Key for the Application 1-3

- Start Android Studio and open the desired project
- Navigate to the Build Menu and select Generate Signed APK
- The Generate Signed APK dialog box is displayed as shown in the following figure:

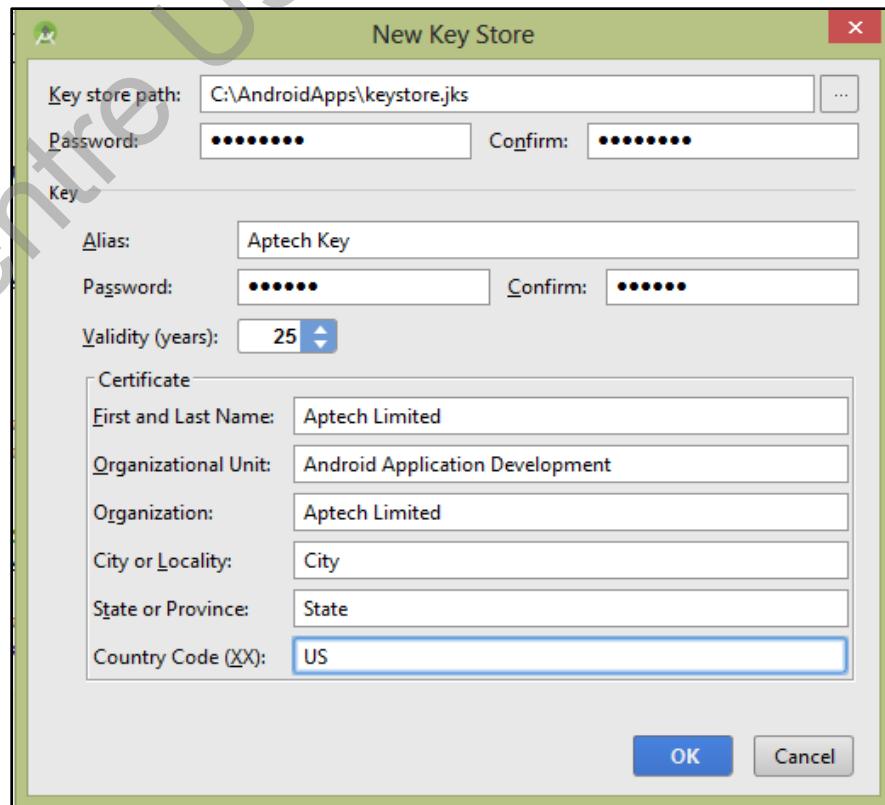


Generating a Key for the Application 2-3

- If it is the first time signing, click Create new to open the New Key Store dialog box as shown in the following figure:



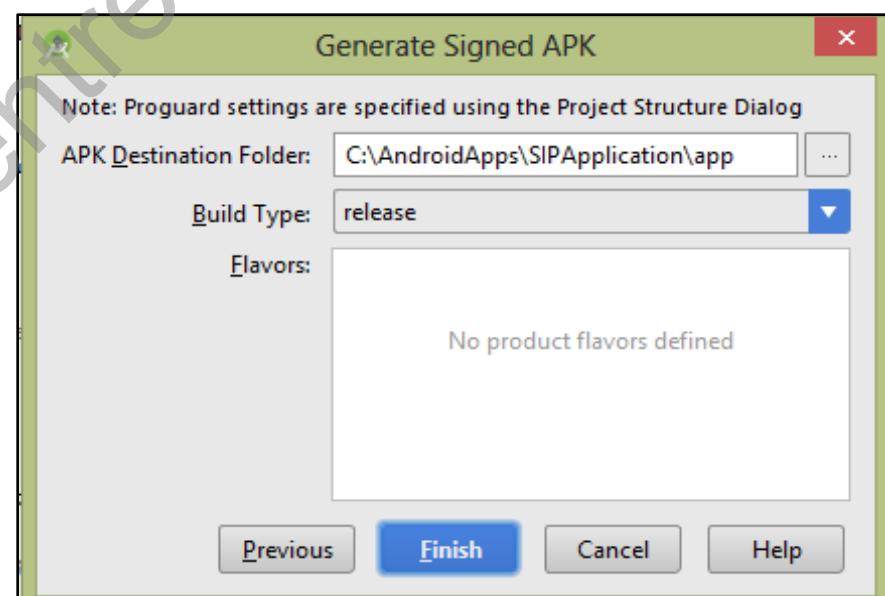
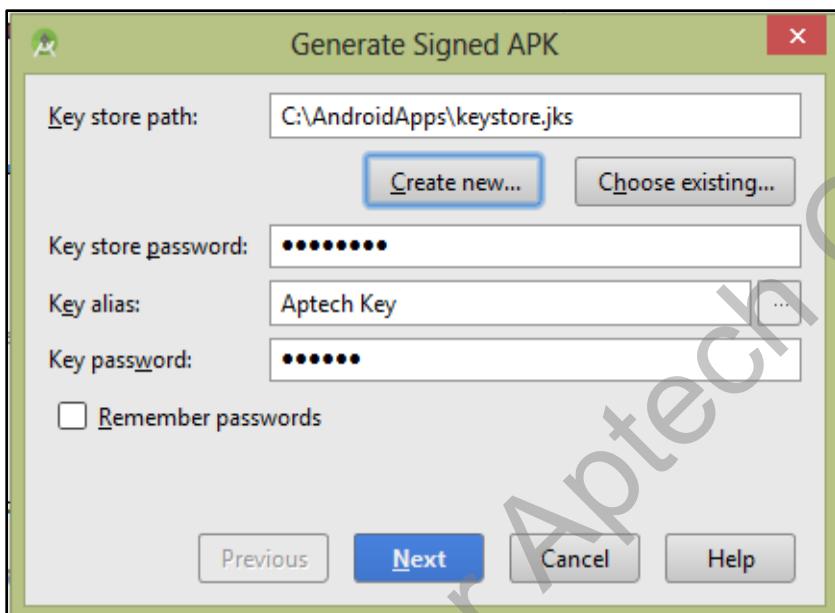
- Enter the details as shown in the following figure:



Generating a Key for the Application 3-3

- Click OK to return to the Generate Signed APK dialog box with the details already entered as shown in the following figure:

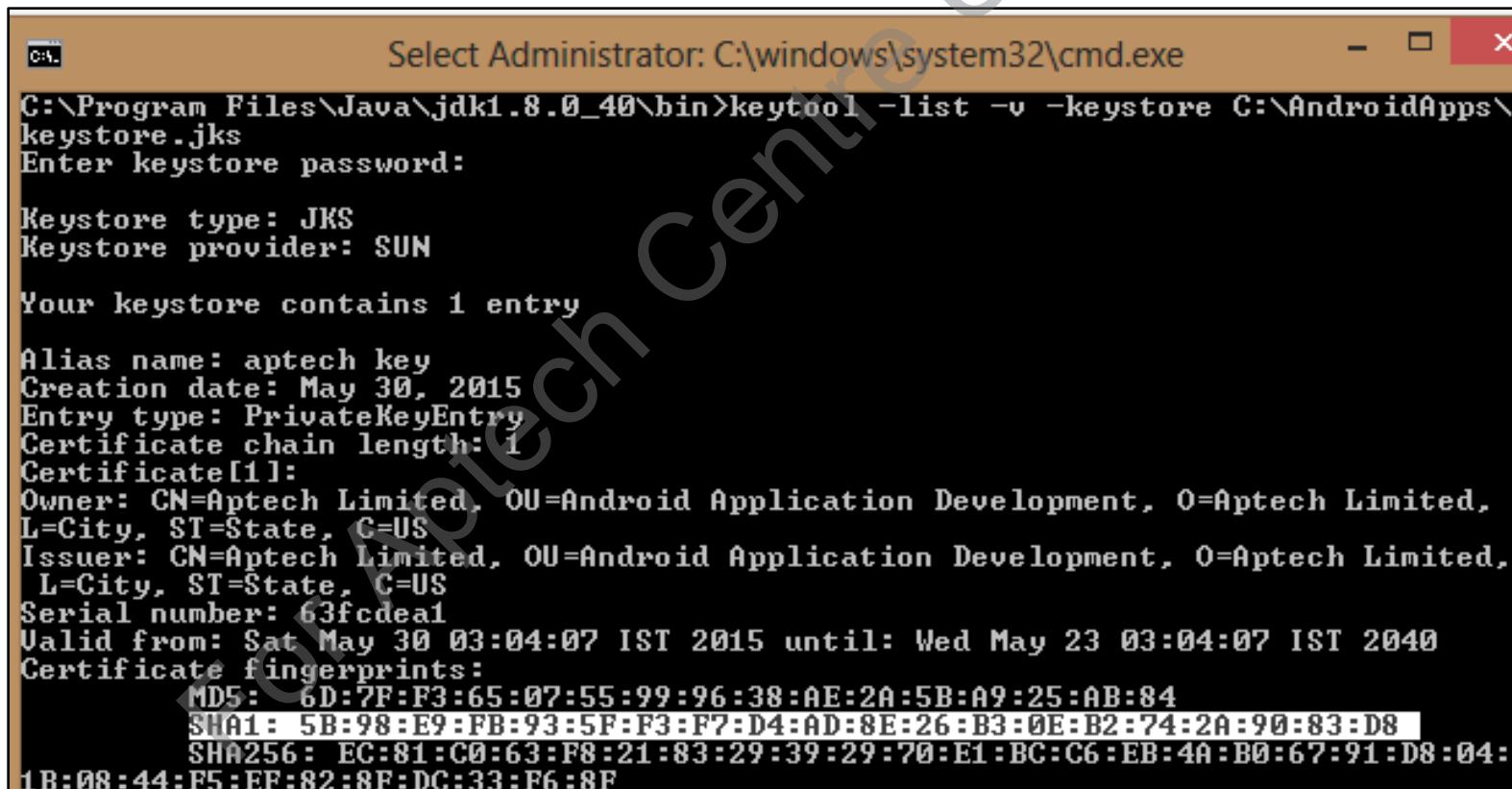
- Click Next to display the Destination dialog box as shown in the following figure:



- Click Finish

Retrieving a Google Maps API Key 1-9

- The keytool.exe is present in the C:\Program Files\Java\<JDK_version_number>\bin folder
- Type the following command to extract the SHA1 fingerprint: keytool -list -v -keystore C:\AndroidApps\keysotre.jks. When prompted, enter the password as shown in the following figure:



```
Select Administrator: C:\windows\system32\cmd.exe
C:\Program Files\Java\jdk1.8.0_40\bin>keytool -list -v -keystore C:\AndroidApps\keystore.jks
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: aptech key
Creation date: May 30, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Aptech Limited, OU=Android Application Development, O=Aptech Limited,
L=City, ST=State, C=US
Issuer: CN=Aptech Limited, OU=Android Application Development, O=Aptech Limited,
L=City, ST=State, C=US
Serial number: 63fcde1
Valid from: Sat May 30 03:04:07 IST 2015 until: Wed May 23 03:04:07 IST 2040
Certificate fingerprints:
    MD5: 6D:2F:F3:65:07:55:99:96:38:AE:2A:5B:A9:25:AB:84
    SHA1: 5B:98:E9:FB:93:5F:F3:F7:D4:AD:8E:26:B3:0E:B2:74:2A:90:83:D8
    SHA256: EC:81:C0:63:F8:21:83:29:39:29:70:E1:BC:C6:EB:4A:B0:67:91:D8:04:
1B:08:44:F5:EF:82:8F:DC:33:F6:8F
```

Retrieving a Google Maps API Key 2-9

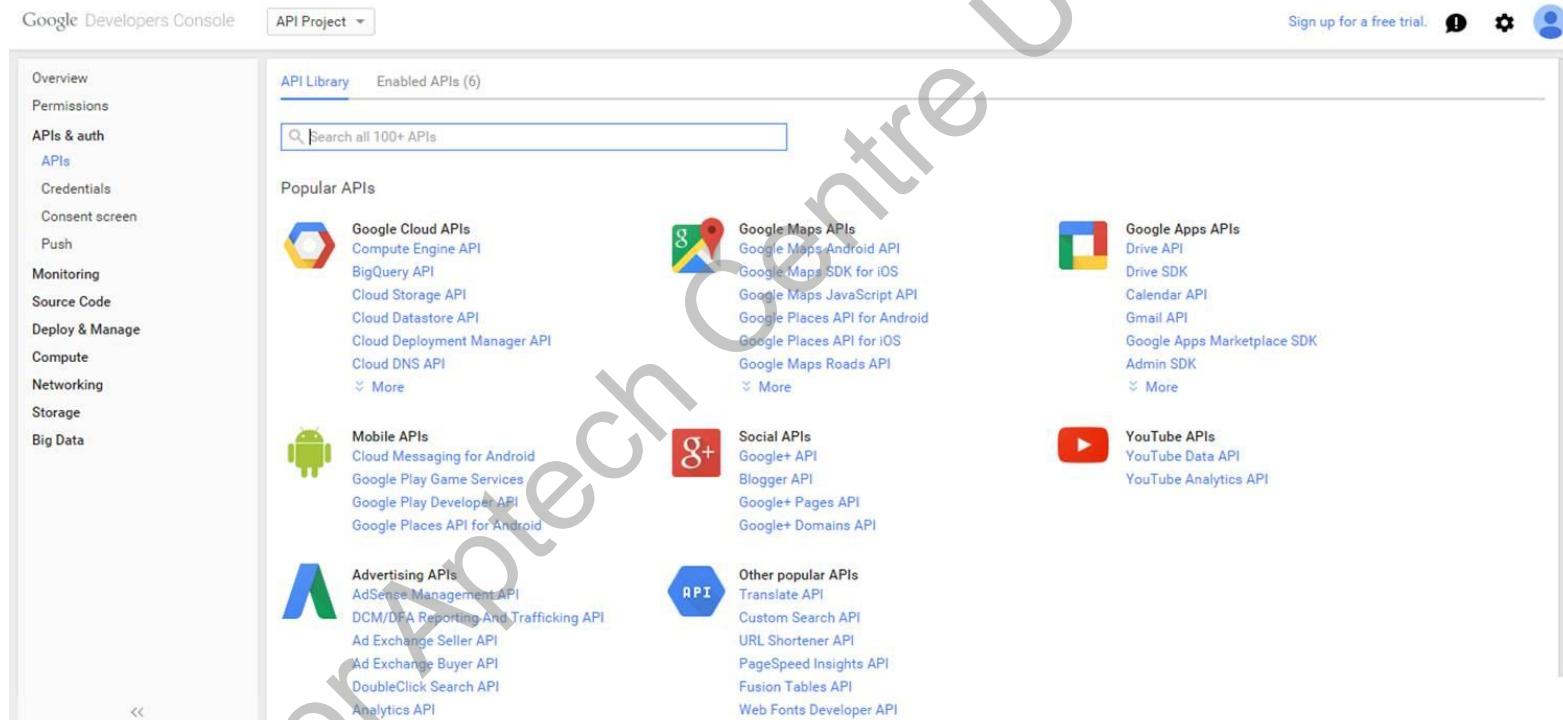
- Enter <https://console.developers.google.com/> in the address bar of a browser
- Sign-in with the Gmail ID of the developer. If you are using the Google APIs Console for the first time, then click Create Project to create a new project named API Project as shown in the following figure:

The screenshot shows the Google Developers Console Project Dashboard for a project named "helical-bonsai-96316". The dashboard features several quickstart guides and management links:

- Project ID:** helical-bonsai-96316 **Project Number:** 505047130652 **Estimated charges this month:** \$0.00
- Project Dashboard:** A central section with three main quickstarts:
 - Take the App Engine quickstart:** Learn "Hello World" for App Engine and deploy your first app to the cloud. Duration: ~10 minutes. [Try App Engine](#)
 - Take the Compute Engine quickstart:** Use Node.js and MongoDB on Compute Engine to create a two-tier to-do list application. Duration: ~15 minutes. [Try Compute Engine](#)
 - Launch click-to-deploy software:** Deploy popular open stacks on Google Compute Engine just by filling out a simple form. [Try Click to Deploy](#)
- APIs & auth:** A sidebar menu with various options like Overview, Permissions, APIs & auth, Monitoring, Source Code, Deploy & Manage, Compute, Networking, Storage, and Big Data.
- Other sections:** Try BigQuery with population data, Create a storage bucket, Boost your app with a Google API, and Enable an API.

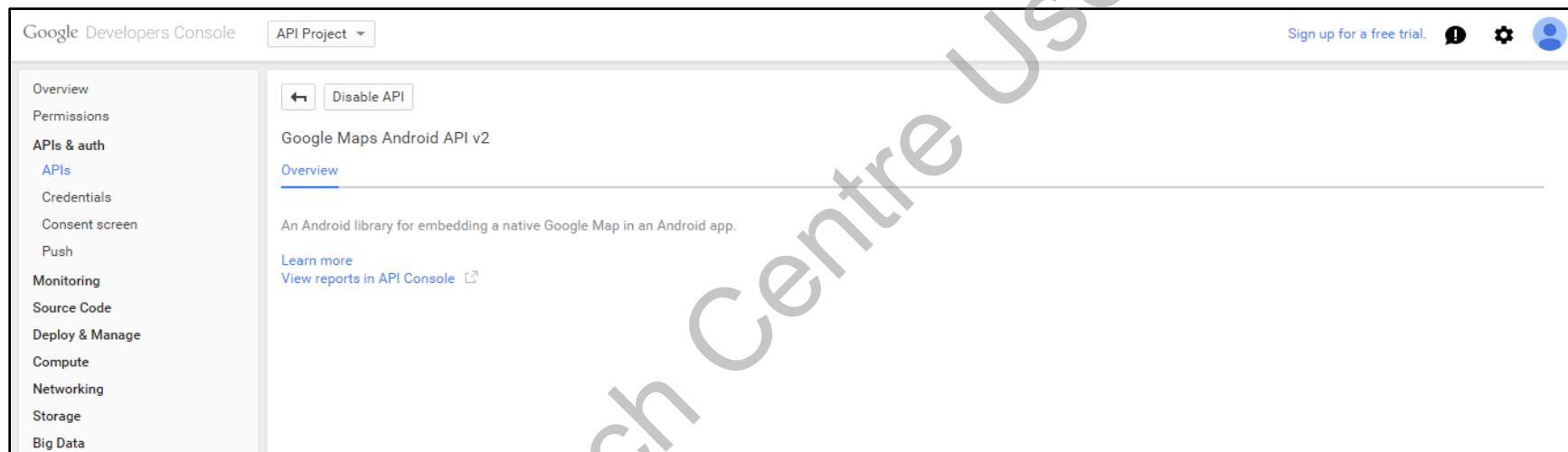
Retrieving a Google Maps API Key 3-9

- Click APIs & auth and select APIs from the left pane as shown in the following figure:



Retrieving a Google Maps API Key 4-9

- Select Google Maps Android API under Google Maps APIs and click Enable API. The output will be as shown in the following figure:



Retrieving a Google Maps API Key 5-9

- Select Google Maps Android API under Google Maps APIs and click Enable API. The output will be as shown in the following figure:

The screenshot shows the Google Developers Console interface. On the left, there is a sidebar with various navigation links: Overview, Permissions, APIs & auth, APIs (which is currently selected and highlighted in blue), Credentials, Consent screen, Push, Monitoring, Source Code, Deploy & Manage, Compute, Networking, Storage, and Big Data. At the top center, it says "API Project" with a dropdown arrow. To the right of that, there are buttons for "Sign up for a free trial.", a profile icon, a gear icon, and a user icon. The main content area is titled "Google Maps Android API v2" and has a "Disable API" button. Below the title, there's a link to "Overview" which is underlined, indicating it's the current page. A descriptive text follows: "An Android library for embedding a native Google Map in an Android app." There are also "Learn more" and "View reports in API Console" links.

Retrieving a Google Maps API Key 6-9

- From the left pane, click Credentials as shown in the following figure:

The screenshot shows the Google Developers Console interface. The top navigation bar includes 'Google Developers Console' (with a dropdown arrow), 'Android API' (selected from a dropdown menu), 'Sign up for a free trial.', and user profile icons. On the far left is a vertical sidebar with various service links: Overview, Permissions, APIs & auth, APIs, **Credentials** (which is the selected item, highlighted in blue), Consent screen, Push, Monitoring, Source Code, Deploy & Manage, Compute, Networking, Storage, and Big Data. The main content area has two sections: 'OAuth' and 'Public API access'. The 'OAuth' section displays the message 'No client IDs found.' and contains a link 'Learn more' and a blue button 'Create new Client ID'. The 'Public API access' section displays the message 'No keys found.' and contains a link 'Learn more' and a blue button 'Create new Key'.

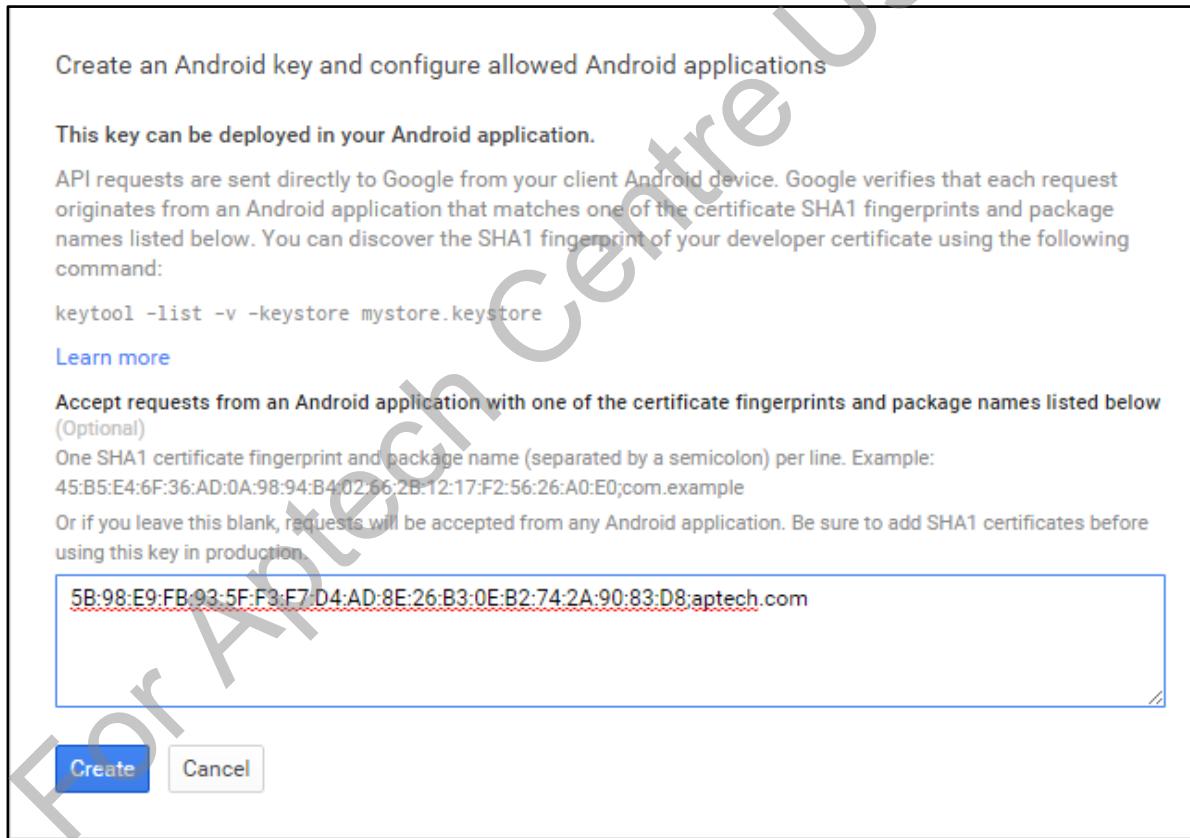
Retrieving a Google Maps API Key 7-9

- Click Create new Key
- Click Android key to obtain an API key from the popup as shown in the following figure:



Retrieving a Google Maps API Key 8-9

- Enter the application key followed by a semicolon and the package name as shown in the following figure:



Retrieving a Google Maps API Key 9-9

- Click Create to generate an API key. The API key is created and displayed on the Credentials page as shown in the following figure:

The screenshot shows the Google Developers Console interface. The left sidebar lists various services: Overview, Permissions, APIs & auth, APIs, Credentials (which is selected and highlighted in blue), Consent screen, Push, Monitoring, Source Code, Deploy & Manage, Compute, Networking, Storage, and Big Data. The main content area has tabs for OAuth and Public API access. Under OAuth, there is a section titled "OAuth" with a note that no client IDs have been found, and a button labeled "Create new Client ID". Under Public API access, there is a section titled "Key for Android applications" which displays the following information:

API key	AlzaSyDr-IOLYKwNOjD8xh5NpNb4_KBckQVM2uY
Android applications	5B:98:E9:FB:93:5F:F3:F7:D4:AD:8E:26:B3:0E:B2:74:2A:90:83:D8,apttech.com
Activation date	Jun 1, 2015, 5:08:00 PM
Activated by	username@gmail.com (you)

At the bottom of this section are three buttons: "Edit allowed Android applications", "Regenerate key", and "Delete".

Google API Manifest Changes 1-2

- ◆ The AndroidManifest.xml needs to be modified to add the key and set the permissions for your application to access Android features and Google Maps servers
- ◆ An example is shown in the following Code Snippet:

```
<permission android:name="com.trafficapp.permission.MAPS_RECEIVE"
    android:protectionLevel="signature"></permission>
<uses-permission android:name="com.trafficapp.permission.MAPS_RECEIVE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-feature android:name="android.hardware.location" android:required="false" />
<uses-feature android:name="android.hardware.location.network" android:required="false" />
<uses-feature android:name="android.hardware.location.gps" />
<uses-feature android:name="android.hardware.wifi" android:required="false" />
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyDr-IOLYKwNOjD8xh5NpNb4_KBckQVM2uY"/>
```

Google API Manifest Changes 2-2

- ◆ If the project is made using the Google Maps activity, a string resource is used instead
- ◆ In this case, navigate to res → values and open google_maps_api.xml for editing and modify the code as shown in the following Code Snippet:

```
<resources>
<string name="google_maps_key" translatable="false"
templateMergeStrategy="preserve">
    AIzaSyDr-IOLYKwNOjD8xh5NpNb4_KBckQVM2uY
</string>
</resources>
```

Map Fragment Example

- The code to add a Map Fragment is shown in the following Code Snippet:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
        <fragment
            android:id="@+id/map"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            class="com.google.android.gms.maps.SupportMapFragment" />
    </LinearLayout>
</RelativeLayout>
```

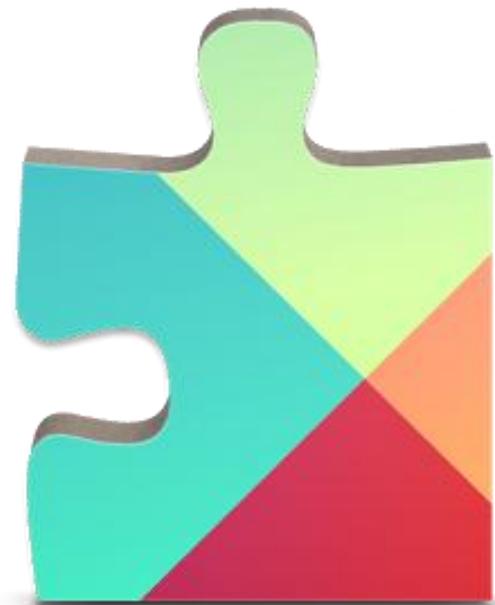
Google Maps Example Application

- Using the code, an application for demonstrating Google Maps is created as shown in the following figure:



Google Play Services

- ◆ Play Services is a client library available for use by other applications to access the Google API
- ◆ Play Services is installed as an APK on devices by default. It is automatically updated on a regular basis
- ◆ On older devices, where play services is not pre-installed, the Play Store will make an automatic install
- ◆ The developer can assume that almost all devices have Play Services installed and freely use the services provided by the library



Google Play Services Classes 1-2

◆ **GoogleApiClient.Builder**

- ❖ The builder class is used to retrieve a reference to a GoogleApiClient object
- ❖ The API that will be utilized by the application can be specified using the addAPI() method
- ❖ The addConnectionCallbacks() method is used to set the callback listener
- ❖ The addOnConnectionFailedListener() method is used to set a call back listener if the connection to the service fails
- ❖ The build() method is used to retrieve an instance of GoogleApiClient

◆ **GoogleApiClient**

- ❖ This class is the main entry point for Google Play Services integration
- ❖ The blockingConnect() and connect() methods are used to make a blocking and a non-blocking connect respectively
- ❖ The disconnect() and reconnect() methods are used to disconnect and reconnect respectively
- ❖ The getSessionId() method is used to get the session id for the connection

Google Play Services Classes 2-2

- ◆ **GoogleApiClient**

- ◆ The registerConnectionCallbacks() is used to register callback listeners for non-blocking connect and unregisterConnectionCallbacks() method is used to unregister the listener

- ◆ **GooglePlayServicesUtil**

- ◆ A utility class which can be used to verify that Google Play Services are available on the devices and to check the version to ensure that they are up to date
 - ◆ The static method isGooglePlayServicesAvailable() is used to check if Google Play Services is installed

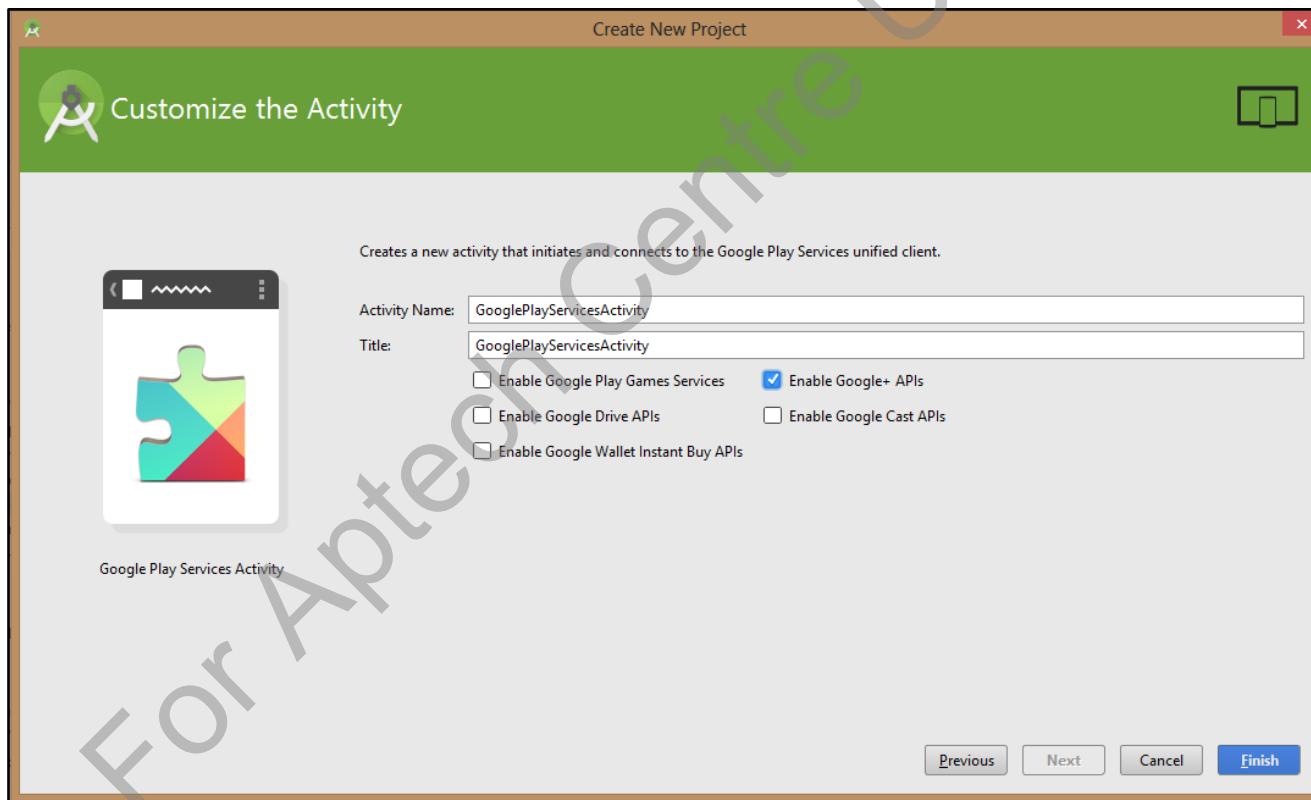
Adding Dependencies for the Application 1-4

- Create a project named PlayServices in Android Studio
- During the activity selection screen, select Google Play Services Activity as shown in the following figure:



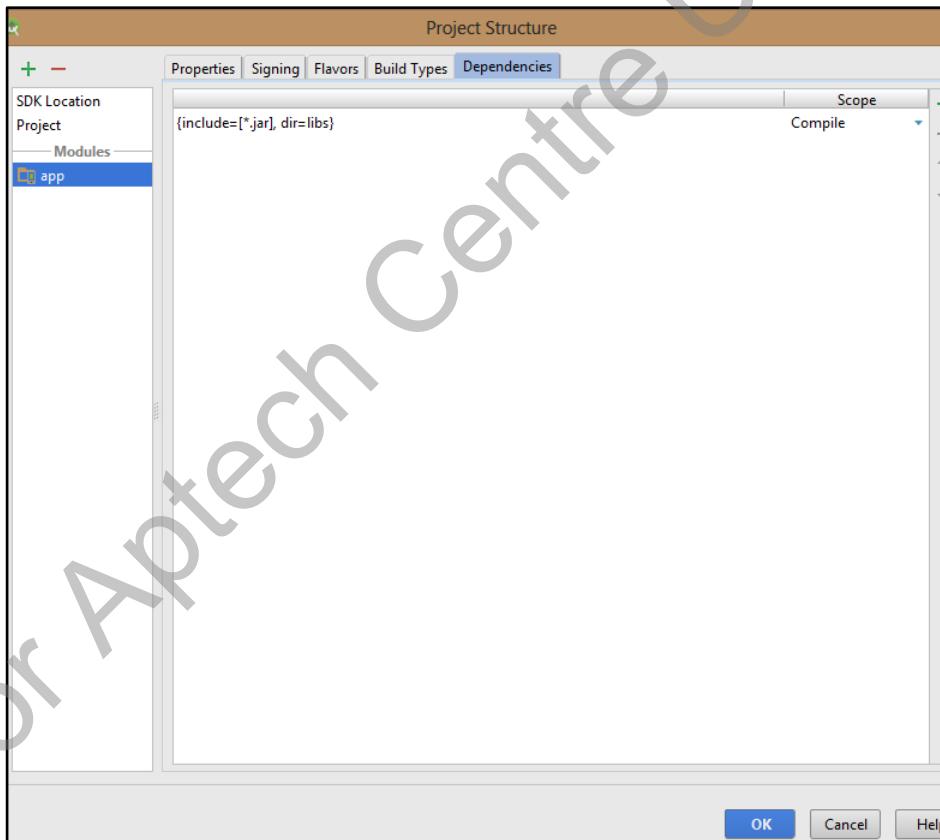
Adding Dependencies for the Application 2-4

- Click Next
- Select the Enable Google + APIs check box as shown in the following figure:



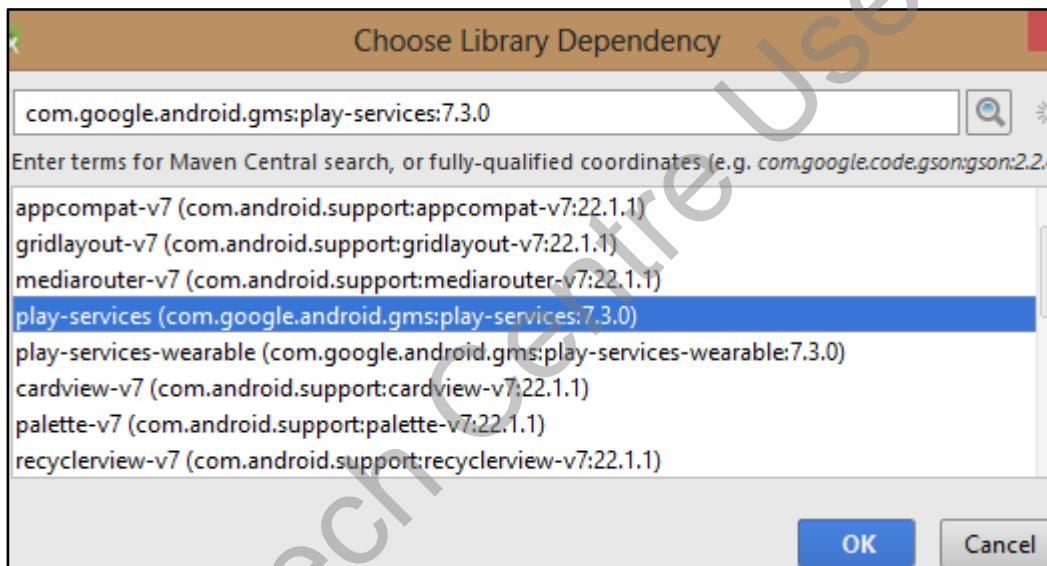
Adding Dependencies for the Application 3-4

- If the project is created using a Blank Activity, right-click app and select Open Module Settings
- From the Project Structure dialog box, select the Dependencies tab as shown in the following figure:



Adding Dependencies for the Application 4-4

- Click the '+' sign and select Library Dependency to open Choose Library Dependency dialog box as shown in the following figure:



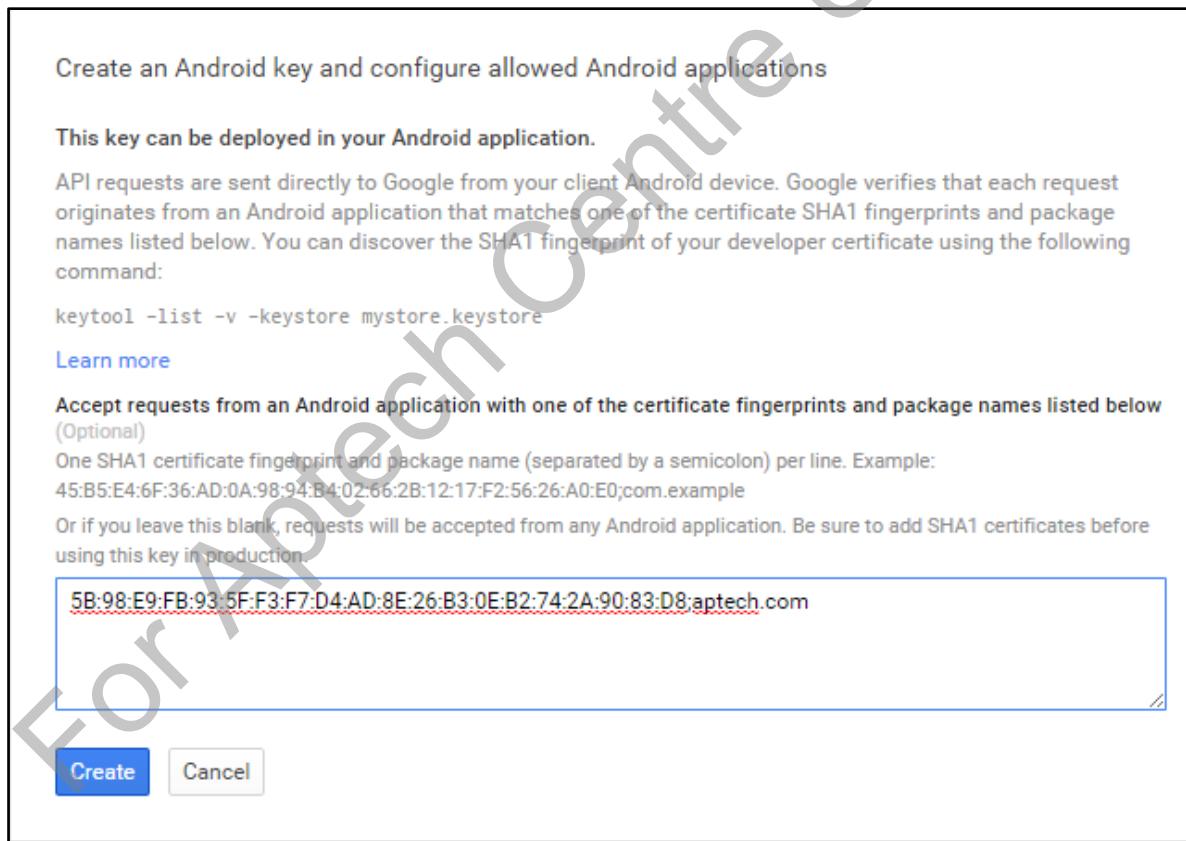
- Select the play-services library and click OK
- Click OK again to confirm the dependencies
- Navigate to app → java → aptech.com.playservices and add the imports
- Generate a Key for the Application

Retrieve a Google+ API Key 1-3

- ◆ The keytool.exe is present in the C:\Program Files\Java\<JDK_version_number>\ bin folder
- ◆ Type the following command to extract the SHA1 fingerprint:
`keytool -list -v -keystore C:\AndroidApps\keysotre.jks` .When prompted, enter the Password
- ◆ Enter <https://console.developers.google.com/> in the Address bar of a browser. This opens the Google console to register your application for the Maps API
- ◆ Sign-in with the Gmail id of the developer. If you are using the Google APIs Console for the first time, then click Create Project to create a new project named API Project
- ◆ Click API & auth and select APIs
- ◆ Select Google+ API and click Enable API
- ◆ From the Left pane, click Credentials

Retrieve a Google+ API Key 2-3

- Click Create new key
- Click Create new Android Key to obtain an API key from the popup
- Enter the application key followed by a semicolon and the package name as shown in the following figure:



Retrieve a Google+ API Key 3-3

- Click Create to generate an API key. The API key is created and displayed on the Credentials page as shown in the following figure:

The screenshot shows the Google Developers Console interface. The left sidebar contains navigation links: Overview, Permissions, APIs & auth, APIs, Credentials (which is selected), Consent screen, Push, Monitoring, Source Code, Deploy & Manage, Compute, Networking, Storage, and Big Data. The main content area has tabs for OAuth and Public API access. Under OAuth, there is a message stating "No client IDs found." and a blue "Create new Client ID" button. Under Public API access, there is a section titled "Key for Android applications" with the following details:

API key	5B:98:E9:FB:93:5F:F3:F7:D4:AD:8E:26:B3:0E:B2:74:2A:90:83:D8;apttech.com
Android applications	AlzaSyDr-IOLYKwNOjD8xh5NpNb4_KBckQVM2uY
Activation date	Jun 1, 2015, 5:08:00 PM
Activated by	chaitanya.nomail@gmail.com (you)

At the bottom of this section are three buttons: "Edit allowed Android applications", "Regenerate key", and "Delete".

Google+ Login Button Example

- The code to add a Google+ Sign in Button is shown in the following Code Snippet:

```
<LinearLayout>

<com.google.android.gms.common.SignInButton
    android:id="@+id/gpluslogin"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onGPlusClicked"
    android:layout_marginBottom="20dp"/>

</LinearLayout>
```

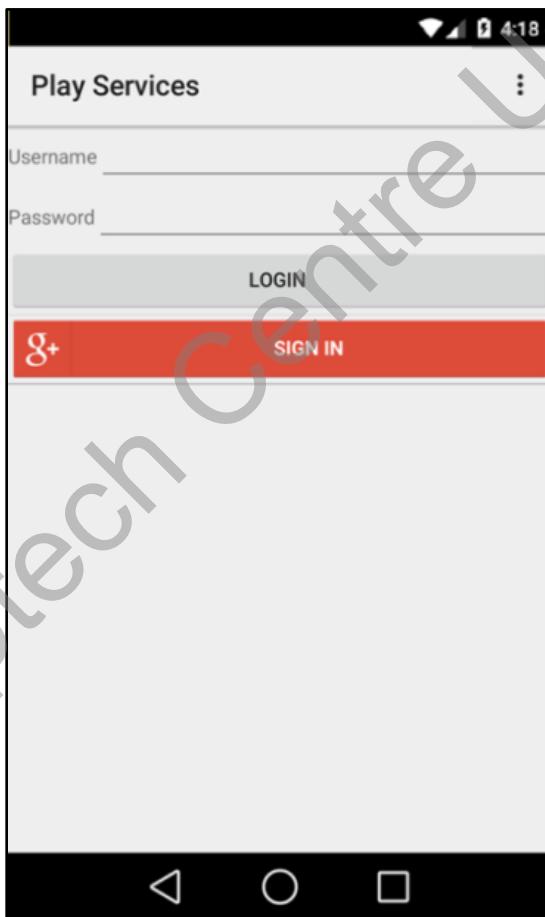
Retrieving User Information Example

- The code to retrieve logged in user information is shown in the following Code Snippet:

```
try {  
    if (Plus.PeopleApi.getCurrentPerson(mGoogleApiClient) != null) {  
        Person user = Plus.PeopleApi.getCurrentPerson(mGoogleApiClient);  
  
        String userName =  
Plus.AccountApi.getAccountName(mGoogleApiClient);  
        String userRealName = user.getDisplayName();  
        String userPhotoUrl = user.getImage().getUrl();  
        String userCoverUrl = user.getCover().getCoverPhoto().getUrl();  
    }  
}
```

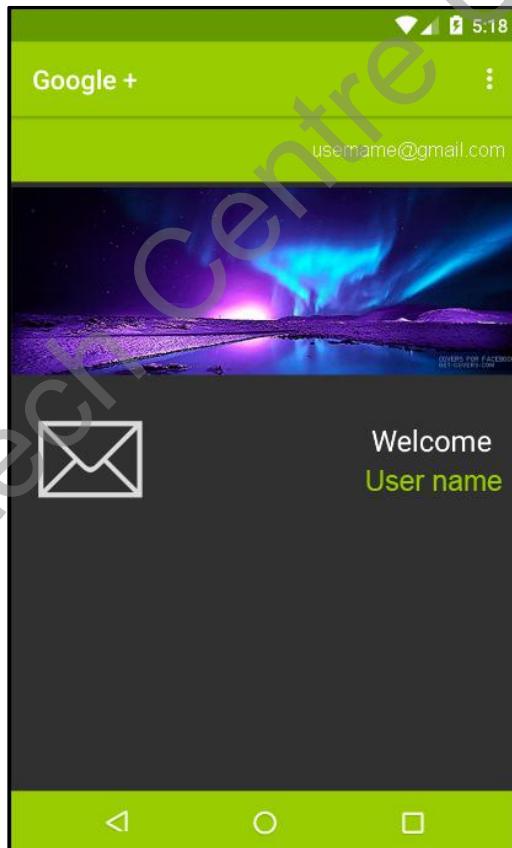
Google API Example 1-3

- Using the code, an application for demonstrating Google API is created as shown in the following figure:



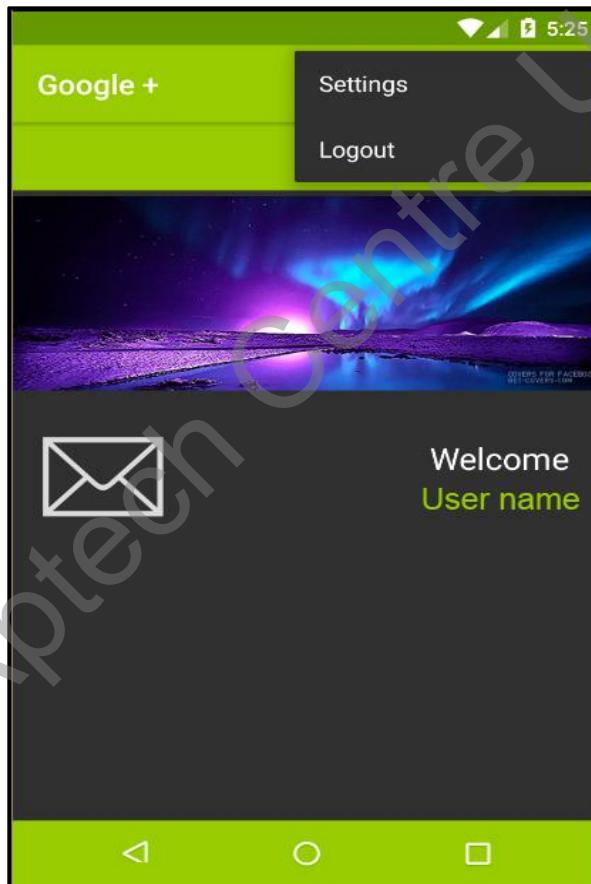
Google API Example 2-3

- Click the Red 'g+' sign in the SIGN IN button and sign in to your Google+ account
- Once the user signs in, the Google+ Home page is displayed as shown in the following figure:



Google API Example 3-3

- The users can logout from the action menu and return to the login page as shown in the following figure:



Google API Application Logic

- ◆ A Google+ Sign in Button is added for the Main Activity
- ◆ An onClick() listener is registered for the Login Button
- ◆ Once the login process is completed, the login listener is triggered
- ◆ The user details are then retrieved from the Login Listener
- ◆ The user details are added as extras and the login Home Page is loaded



Summary

- ◆ Google provides API for developers to access their product services such as search, drive, App Engine and so on
- ◆ Location-based Services (LBS) refer to a set of applications that exploit the knowledge of the geographical position of a mobile device in order to provide services based on that information
- ◆ Google Maps allow the developers to develop applications of world with rich maps provided by Google
- ◆ Google API Add on is an extension to Android SDK development environment that lets the developer develop applications for devices that include Google's set of custom applications, libraries, and services
- ◆ The GoogleAPIClient and GooglePlayServicesUtil classes are used to access Google API
- ◆ Google Play Services is a client side library for accessing Google services