



Mobile Apps Development

COMP-304
Fall 2018



Review of Lecture 7

❑ Content Providers

- A **data** store that applications can query, edit, add, and delete.
- The data can be stored in a **database**, in **files**, or even over a **network**

The format of the query URI is as follows:

```
<standard_prefix>://<authority>  
>/<data_path>/<id>  
  
content://contacts/people/2
```

❑ Creating a Content Provider

- Create a class that extends the **ContentProvider**
- Override methods:
 - query
 - getType()
 - insert
 - delete
 - update
 - onCreate
- ❑ You can use an SQLite database or other APIs.



Review of Lecture 7

- ❑ Use an **UriMatcher** object to parse the content URI that is passed to the content provider through a **ContentResolver**

- ❑ **Register your content provider** with Android:

- modify the `AndroidManifest.xml` file by adding the `<provider>` element

- ❑ **Using the Content Provider**

- Create and populate `ContentValues` object
- Call `ContentProvider` methods on `ContentResolver` object:

```
Uri uri =  
    getContentResolver().insert(  
        Uri.parse(  
            "content://net.learn2develop.  
            provider.Books/books  
            "),  
        values);
```



Location-Based Services

Objectives:

- ❑ Develop Android Apps with GPS and Maps capabilities
 - Displaying **Google Maps** in your Android application
 - Displaying **zoom controls** on the map
- ❑ Use GPS Features in Your Applications for finding your location, locating your emulator, geocoding locations, mapping locations, mapping intents, mapping views
 - Switching between the **different map views**
 - Retrieving the **address location** touched on the map
 - Performing **geocoding** and **reverse geocoding**



Displaying Maps

- ❑ Google Maps is one of the many applications bundled with the Android platform
- ❑ You can **embed a map** into your own applications and use for different purposes
- ❑ The steps:
 1. Install the Google Play services SDK
 2. Create a Google Maps project
 3. Get a Google Maps API key



Getting a Maps API Key

- ❑ Use the link provided in the `google_maps_api.xml` file that Android Studio created for you:
- ❑ Copy the link provided in the `google_maps_api.xml` file and paste it into your browser.
- ❑ The link takes you to the Google Cloud Platform Console and supplies the required information to the Google Cloud Platform Console via URL parameters, thus reducing the manual input required from you.
- ❑ Follow the instructions to **create a new project** on the Google Cloud Platform Console **or select an existing project**.



Getting a Maps API Key

Register your application for Maps SDK for Android 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.

My Project

Terms of Service

☒ I have read and agree to the [GCP Marketplace Terms of Service](#).

Country of residence

Canada

I would like to receive periodic emails on news, product updates and special offers from Google Cloud and Google Cloud Partners.

☐ Yes ☒ No

Agree and continue



Getting a Maps API Key

The API is enabled

Maps SDK for Android has been enabled.

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

Create API key

Credentials

API key created

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

Your API key

AIzaSyA1qr...JSZ3y...JLwa34RuYcVYcs...rsU



⚠ Restrict your key to prevent unauthorized use in production.

CLOSE

RESTRICT KEY



Adding the API Key to your application

❑ To add the key to your application:

- In AndroidManifest.xml, add the following element as a child of the `<application>` element, by inserting it just before the closing tag `</application>`:

`<meta-data`

```
android:name="com.google.android.geo.API_KEY"  
android:value="API_KEY
```

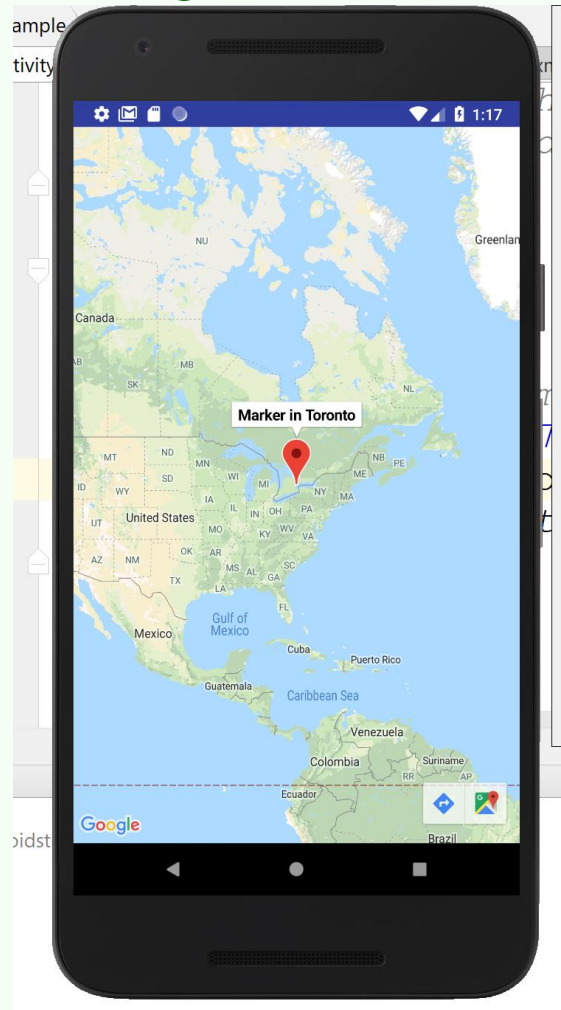
substituting your API key for *API_KEY*.

- This element sets the key `com.google.android.geo.API_KEY` to the value *API_KEY* and makes the API key visible to any **MapFragment** in your application.



Running the app

- ❑ Change the Latitude, Longitude to Toronto values
- ❑ Run the app





Adding built-in zoom control

- ❑ To add a parameter to `activity_maps.xml` that sets the **uiZoomControls to true**:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    map:uiZoomControls="true"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

- ❑ You can also programmatically zoom in or out of the map using the `animateCamera()` method of the `GoogleMap` class.



Programmatically zoom in or out

```
public boolean onKeyDown(int keyCode, KeyEvent event) {  
    switch (keyCode) {  
        case KeyEvent.KEYCODE_3:  
            mMap.animateCamera(CameraUpdateFactory.zoomIn());  
            break;  
        case KeyEvent.KEYCODE_1:  
            mMap.animateCamera(CameraUpdateFactory.zoomOut());  
            break;  
    }  
    return super.onKeyDown(keyCode, event);  
}
```



Changing Views

- ❑ By default, Google Maps is displayed in *map view*, which is basically drawings of streets and places of interest.
- ❑ You can also set Google Maps to display in *satellite view* using the `setMapType()` method of the `GoogleMap` class:

```
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));  
    mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);  
}
```



Getting the Location That Was Touched

- ❑ To get the latitude and longitude of a point on the Google Map that was touched, you must set a `onMapClickListener`:

```
mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener()
{
    @Override
    public void onMapClick(LatLng point) {
        Log.d("DEBUG", "Map clicked [" + point.latitude +
            " / " + point.longitude + "]);
    }
});
```



Reverse Geocoding

- ❑ Google Maps in Android supports **reverse geocoding** via the **Geocoder** class.
- ❑ The following code snippet shows how you can **retrieve the address of a location just touched** using the `getFromLocation()` method:

```
Geocoder geoCoder = new Geocoder(
    getBaseContext(), Locale.getDefault());
try {
    List<Address> addresses = geoCoder.getFromLocation(point.latitude,point.longitude,1);
    String add = "";
    if (addresses.size() > 0)
    {
        for (int i=0; i<addresses.get(0).getMaxAddressLineIndex();i++)
            add += addresses.get(0).getAddressLine(i) + "\n";
    }
    Toast.makeText(getBaseContext(), add, Toast.LENGTH_SHORT).show();
}
catch (IOException e) { e.printStackTrace();}
}
```



Geocoding

- ❑ If you know the address of a location but want to know its **latitude and longitude**, you can do so via **geocoding**.
- ❑ You can use the Geocoder class for this purpose.
- ❑ The following code shows how you can find the exact location of the Empire State Building by using the `getFromLocationName()` method:

```
Geocoder geoCoder = new Geocoder(
    getBaseContext(), Locale.getDefault());
try {
    List<Address> addresses = geoCoder.getFromLocationName("empire state
    building", 5);
    if (addresses.size() > 0) {
        LatLng p = new LatLng((int) (addresses.get(0).getLatitude()),
        (int) (addresses.get(0).getLongitude()));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(p));
    }
} catch (IOException e) {e.printStackTrace();}
```

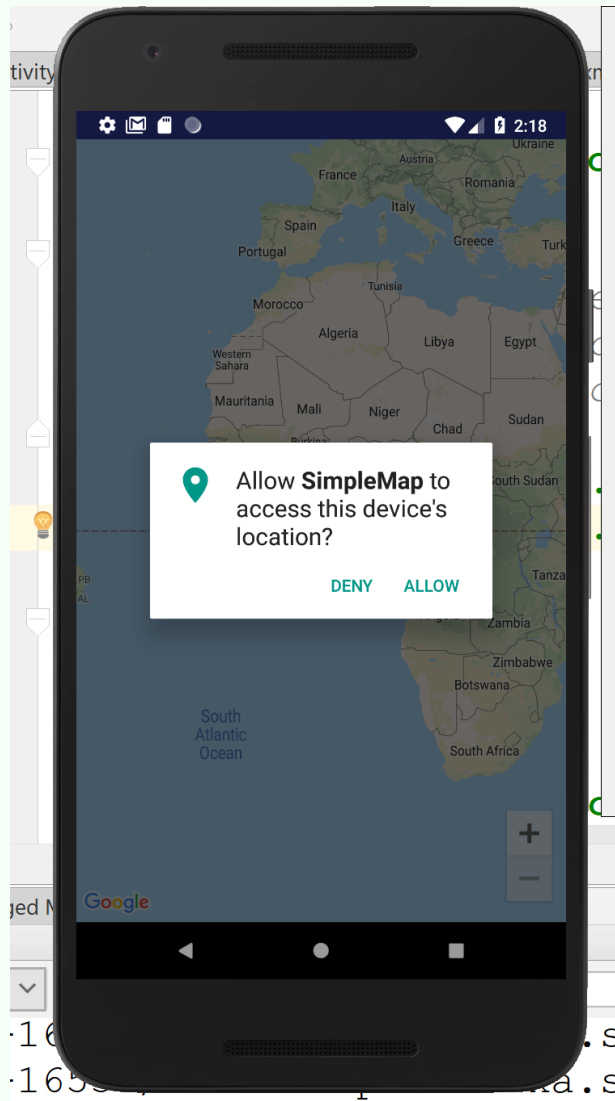



Getting Location Data

- ❑ Mobile devices are commonly equipped with **GPS receivers**.
- ❑ Because of the many satellites orbiting the earth, you can **use a GPS receiver to find your location easily**.
- ❑ However, GPS requires a clear sky to work and hence **does not always work indoors** or where satellites can't penetrate (such as a tunnel through a mountain).
- ❑ Another effective way to locate your position is through *cell tower triangulation*.
 - When a mobile phone is switched on, it is constantly in contact with base stations surrounding it.
 - By knowing the **identity of cell towers**, it is possible to **translate this information into a physical location** through the use of various databases containing the cell towers' identities and their exact geographical locations.



Getting Location Data





Getting Location Data

- ❑ In Android, location-based services are provided by the **LocationManager** class, located in the **android.location** package.
- ❑ Using the **LocationManager** class, your application can obtain periodic updates of the device's geographical locations, as well as fire an intent when it enters the proximity of a certain location.



Getting Location Data

- ❑ In the MapsActivity.java file, you first check for permission to use the Course Locations.
- ❑ Then you obtain a reference to the LocationManager class using the getSystemService() method. You do this in the onCreate() method of the LBSActivity:

//---use the LocationManager class to obtain locations data

```
lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
locationListener = new MyLocationListener();
```



Getting Location Data

- ❑ Next, you create an instance of the MyLocationListener class, which you define later in the class.
- ❑ The MyLocationListener class implements the LocationListener abstract class.
- ❑ You need to override four methods in this implementation:
 1. onLocationChanged(Location location) - Called when the location has changed
 2. onProviderDisabled(String provider) - Called when the provider is disabled by the user
 3. onProviderEnabled(String provider) - Called when the provider is enabled by the user
 4. onStatusChanged(String provider, int status, Bundle extras) - Called when the provider status changes



Getting Location Data

- ❑ In this example, you're more interested in **what happens when a location changes**, so you write your code in the `onLocationChanged()` method.
- ❑ Specifically, when a location changes, you **display a small dialog on the screen showing the new location information**: latitude and longitude.
- ❑ You show this dialog using the Toast class:

```
public void onLocationChanged(Location loc) {  
    if (loc != null) {  
        Toast.makeText(getBaseContext(),  
            "Location changed : Lat: " + loc.getLatitude() + " Lng: " + loc.getLongitude(),  
            Toast.LENGTH_SHORT).show();  
        LatLng p = new LatLng((int) (loc.getLatitude()),(int) (loc.getLongitude()));  
        mMap.moveCamera(CameraUpdateFactory.newLatLng(p));  
        mMap.animateCamera(CameraUpdateFactory.zoomTo(7));  
    }  
}
```



Getting Location Data

- ❑ To be notified whenever there is a change in location, you needed to register a request for location
- ❑ changes so that your program can be notified periodically.
- ❑ You do this via the `requestLocationUpdates()` method:

```
if(permissionGranted) {  
    lm.requestLocationUpdates(LocationManager.GP  
        S_PROVIDER, 0, 0, locationManager);  
}
```



Getting Location Data

- ❑ The `requestLocationUpdates()` method takes four arguments:
 1. `provider` - The name of the provider with which you register.
 - In this case, you are using GPS to obtain your geographical location data.
 2. `minTime` - The minimum time interval for notifications, in milliseconds. 0 indicates that you want to be continually informed of location changes.
 3. `minDistance` - The minimum distance interval for notifications, in meters. 0 indicates that you want to be continually informed of location changes.
 4. `listener` - An object whose `onLocationChanged()` method will be called for each location update



Getting Location Data

- ❑ Finally, in the `onPause()` method, you remove the listener when the activity is destroyed or goes into the background (so that the application no longer listens for changes in location, thereby saving the battery of the device).
- ❑ You do that using the `removeUpdates()` method:



Getting Location Data

```
@Override
public void onPause() {
    super.onPause();
    //---remove the location listener---
    if (ActivityCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
            this, android.Manifest.permission.ACCESS_COARSE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this,
            new String[]{android.Manifest.permission.ACCESS_COARSE_LOCATION},
            REQUEST_COARSE_ACCESS);
        return;
    }else{
        permissionGranted = true;
    }
    if(permissionGranted) {
        lm.removeUpdates(locationListener);
    }
}
```



Getting Location Data

- ❑ You can combine both the GPS location provider with the network location provider within your application:

@Override

```
public void onResume() {  
    super.onResume();  
    //---request for location updates---  
    lm.requestLocationUpdates(  
        LocationManager.GPS_PROVIDER, 0, 0, locationListener);  
    //---request for location updates---  
    lm.requestLocationUpdates(  
        LocationManager.NETWORK_PROVIDER, 0, 0,  
        locationListener);  
}
```



Getting Location Data

- ❑ However, this will cause your application to receive two different sets of coordinates, as both the GPS provider and the NETWORK provider will try to get your location using their own methods (GPS versus Wi-Fi and cell ID triangulation).
- ❑ Hence, it is important that you **monitor the status of the two providers in your device and use the appropriate one.**
- ❑ You can **check the status of the two providers by implementing the following three methods** (shown in bold) of the MyLocationListener class:



Getting Location Data

//---called when the provider is disabled---

```
public void onProviderDisabled(String provider) {  
    Toast.makeText(getBaseContext(), provider + " disabled",  
        Toast.LENGTH_SHORT).show();  
}
```

//---called when the provider is enabled---

```
public void onProviderEnabled(String provider) {  
    Toast.makeText(getBaseContext(), provider + " enabled", Toast.LENGTH_SHORT).show();  
}
```

//---called when there is a change in the provider status---

```
public void onStatusChanged(String provider, int status, Bundle extras) {  
    String statusString = "";  
    switch (status) {  
        case android.location.LocationProvider.AVAILABLE:  
            statusString = "available";  
        case android.location.LocationProvider.OUT_OF_SERVICE:  
            statusString = "out of service";  
        case android.location.LocationProvider.TEMPORARILY_UNAVAILABLE:  
            statusString = "temporarily unavailable";  
    }  
    Toast.makeText(getBaseContext(), provider + " " + statusString, Toast.LENGTH_SHORT).show();  
}
```



Monitoring a Location

- ❑ One very cool feature of the `LocationManager` class is its ability to monitor a specific location.
- ❑ This is achieved using the `addProximityAlert()` method.
- ❑ The following code snippet shows how to monitor a particular location such that if the user is within a five-meter radius from that location, your application will fire an intent to launch the web browser:



Monitoring a Location

```
import android.app.PendingIntent;
import android.content.Intent;
import android.net.Uri;
//---use the LocationManager class to obtain locations data---
lm = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
//---PendingIntent to launch activity if the user is within
// some locations---
PendingIntent pendingIntent = PendingIntent.getActivity(
    this, 0, new Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://www.amazon.com")), 0);
lm.addProximityAlert(37.422006, -122.084095, 5, -1, pendingIntent);
```



Monitoring a Location

- ❑ The `addProximityAlert()` method takes **five arguments**:
 - Latitude
 - Longitude
 - Radius (in meters)
 - Expiration (duration for which the proximity alert is valid, after which it is deleted; `-1` for no expiration)
 - Pending intent
- ❑ Note that if the Android device's screen goes to sleep, the proximity is also checked once every four minutes in order to preserve the battery life of the device.



References

❑ Textbook

- Android Documentation:
- <https://developers.google.com/maps/documentation/android-sdk/start>
- <https://developer.android.com/guide/topics/location/index.html>
- <https://developer.android.com/guide/topics/location/strategies.html>
- <https://developer.android.com/training/location/>
- <https://developers.google.com/maps/documentation/android-api/>
- <https://developers.google.com/maps/documentation/android-api/start>
- <https://developers.google.com/maps/documentation/javascript/geolocation>
- <https://developers.google.com/maps/faq>