UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

disim

Mobile Programming Laboratory

ANDROID
Location Services

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

- Location
  - Sensor
  - Google Location Services

Location Services - Sensor

To develop a location-aware app for Android, you can utilize GPS and Android's Network Location Provider to acquire the user location.

GPS pro:

- it is more accurate.

GPS against:

- it only works outdoors;
- it quickly consumes battery power;
- it does not return the location as quickly as users want.

NLP determines user location using cell tower and WiFi signals.

Location Services - Sensor

Getting user location in Android works by means of callback.

You indicate that you'd like to receive location updates from the LocationManager by calling requestLocationUpdates(), passing a LocationListener.

You can control the frequency at which your listener receives updates with the second and third parameter

- min time interval between notifications
- min change in distance between notifications

The request location updates from the GPS provider, use GPS_PROVIDER instead of NETWORK_PROVIDER.

Location Services - Sensor

The following code shows the use of location sensor

```
LocationListener listener = new LocationListener() {
  @Override
  public void onLocationChanged(Location location) {
     // Current fix
  }

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {}

  @Override
  public void onProviderEnabled(String provider) {}

  @Override
  public void onProviderDisabled(String provider) {}
};

LocationManager manager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
if(manager != null)
manager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
```

You must declare the Manifest's permission

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-feature android:name="android.hardware.location.gps" />
```

If the target of app is Android 6.0 (API 23) or higher, you must remember to declare the **run-time permission**

Location Services - Google Location Services

The location API available in Google Play Services facilitate adding location awareness to your app with automated location tracking, geofencing, and activity recognition.

Google encourages to switch to the Google Play services location APIs as soon as possible, if you use in your app the framework location APIs.

To use this APIs you must include the dependency in build.gradle file

```
implementation 'com.google.android.gms:play-services-location:15.0.1'
```

You must declare the permission on Manifest

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-feature android:name="android.hardware.location.gps" />
```

Location Services - Google Location Services

To use it, you must connect to Google Play Services and request the Location API

```
client = new GoogleApiClient.Builder(context)
      .addConnectionCallbacks(connectionCallbacks)
      .addOnConnectionFailedListener(connectionFailed)
      .addApi(LocationServices.API)
      .build();
```

Define the connection callbacks and the failure listener

```
GoogleApiClient.ConnectionCallbacks connectionCallbacks = new GoogleApiClient.ConnectionCallbacks() {
  @Override
  public void onConnected(@Nullable Bundle bundle) {}

  @Override
  public void onConnectionSuspended(int i) {}
};

GoogleApiClient.OnConnectionFailedListener connectionFailed = new GoogleApiClient.OnConnectionFailedListener() {
  @Override
  public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {}
};
```

Now you can connect the client on onStart() method

```
public void onStart(){
  client.connect();
}

public void onStop(){
  client.disconnect();
}
```

Location Services - Google Location Services

To get last known location

```
GoogleApiClient.ConnectionCallbacks connectionCallbacks = new GoogleApiClient.ConnectionCallbacks() {
  @Override
  public void onConnected(@Nullable Bundle bundle) {
     FusedLocationProviderClient locationClient = LocationServices.getFusedLocationProviderClient(context);
     locationClient.getLastLocation()
           .addOnSuccessListener(new OnSuccessListener<Location>() {
              @Override
              public void onSuccess(Location location) {
                 if(location != null) {
                    System.out.println(location.getLatitude() + ", " + location.getLongitude());
                 }
              }
           });
  }

  @Override
  public void onConnectionSuspended(int i) {}
};
```

Location Services - Google Location Services

If your app needs to request location or receive permission update, your app must specify the required level of accuracy/power consumption and desired update interval.

To do this you can use LocationRequest data object.

```
LocationRequest request = new LocationRequest();
request.setInterval(10000);
request.setFastestInterval(5000);
request.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

setInterval(): sets the rate in milliseconds at which your app prefers to receive location updates.

setFastestInterval(): sets the fastest rate in milliseconds. You need to set this rate because other apps also affect the rate at which updates are sent.

setPriority(): sets the priority of the request. The supported values are:

- **PRIORITY_BALANCED_POWER_ACCURACY** - ~100m
- **PRIORITY_HIGH_ACCURACY** - the most precise location possible
- **PRIORITY_LOW_POWER** - ~10Km
- **PRIORITY_NO_POWER** - you want receive location updates when available, if other apps request it

Location Services - Google Location Services

One you have created a LocationRequest, you can add the location request and check whether the current location settings are satisfied:

```
LocationSettingsRequest settings = new LocationSettingsRequest.Builder()
      .addLocationRequest(request)
      .build();

LocationServices.getSettingsClient(context).checkLocationSettings(settings)
      .addOnCompleteListener(new OnCompleteListener<LocationSettingsResponse>() {
         @Override
         public void onComplete(@NonNull Task<LocationSettingsResponse> task) {
            try{
               LocationSettingsResponse response = task.getResult(ApiException.class);
            }catch (ApiException e) {
               e.printStackTrace();
            }
         }
      });
```

Location Services - Google Location Services

Now you can invoke the request for location updates.

```
LocationCallback locationCallback = new LocationCallback(){
  @Override
  public void onLocationResult(LocationResult locationResult) {
     super.onLocationResult(locationResult);
     Location location = locationResult.getLastLocation();
  }
};

FusedLocationProviderClient client = LocationServices.getFusedLocationProviderClient(context);
client.requestLocationUpdates(request, locationCallback, Looper.myLooper());
```

To stop the location updates

```
FusedLocationProviderClient client = LocationServices.getFusedLocationProviderClient(context);
client.removeLocationUpdates(locationCallback);
```

Be Careful! When you use the Google Play Services, your app has to check if the Play Services are available.

```
boolean isAvailable = GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(context) == ConnectionResult.SUCCESS;
```