Développement Android : Géolocalisation

Thierry Vaira <tvaira@free.fr> http://tvaira.free.fr/

25/11/2017 (rev. 1)

Table des matières

Géolocalisation sous Android **1**
Objectif . . . 1
Introduction . . . 1
Les permissions . . . 3
Le gestionnaire de position . . . 3
L'interface LocationListener . . . 4
La classe Geocoder . . . 5
Exemple . . . 7
Liens . . . 11

Géolocalisation sous Android

Objectif

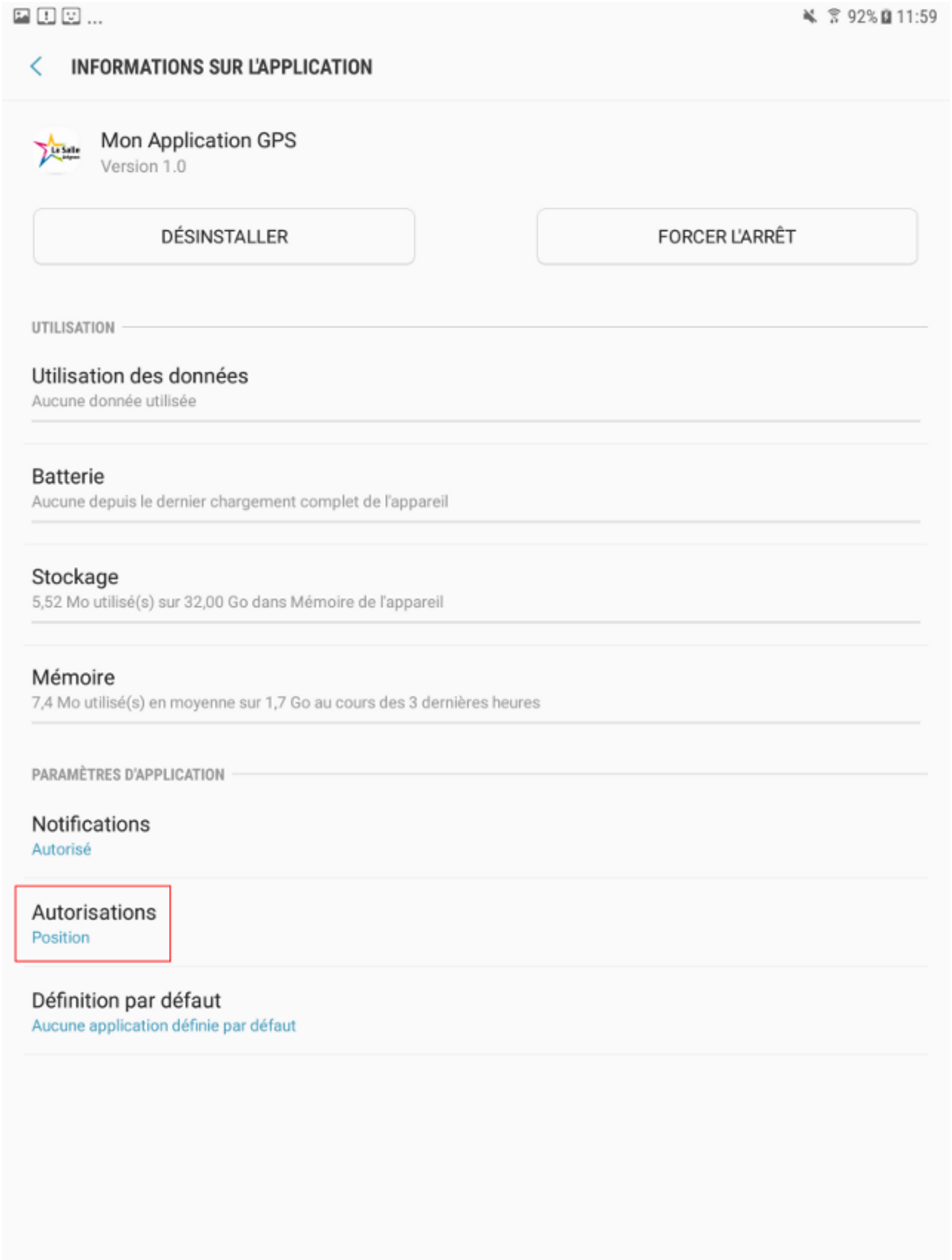
L'objectif est de géolocaliser l'équipement avec assez de précision en utilisant les services de géolocalisation d'Android qui fournit une position géographique selon une latitude et une longitude.

Introduction

Le fournisseur de localisation (*location provide*) propose deux types de géolocalisation :

- la première basée sur le récepteur GPS avec une précision de l'ordre d'une dizaine de mètres (*gps provider*)
- une seconde basée sur le réseau cellulaire et les signaux wifi d'une précision très variable d'une dizaine de mètres à une centaine de mètres (*network provider*)

Le système Android donne un accès de localisation aux applications à condition que les paramètres de sécurité les y autorisent : Paramètres de configuration -> Accès aux données de localisation.



 © 2017 tv <tvaira@free.fr>

Les outils de localisation dans le *package* android.location.

Les permissions

L'application a besoin des droits d'accès (`uses-permission`) pour utiliser les fonctionnalités de localisation :

- une géolocalisation par GPS : `ACCESS_FINE_LOCATION` ;
- une localisation par WiFi et antennes relais : `ACCESS_COARSE_LOCATION` ;

Pour cela, il faut modifier le fichier `AndroidManifest.xml` de l'application :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.tv.myapplication5">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    ...
</manifest>
```

Le gestionnaire de position

Il faut tout d'abord disposer d'un gestionnaire de position à partir de la classe LocationManager.

On le récupère de la manière suivante :

```
LocationManager locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
```

Il existe plusieurs méthodes pour récupérer les fournisseurs de position disponibles sur le terminal mobile :

- `getAllProviders()` qui permet de récupérer une liste de tous les noms de fournisseur (même si vous n'avez pas les droits d'accès pour certains) ;
- `getProviders(boolean enabledOnly)` qui permet de récupérer une liste de tous les noms de fournisseur accessible ;
- `getProvider(String name)` qui retourne un `LocationProvider` à partir de son nom.

Il est aussi possible d'obtenir le fournisseur à partir de critères à déterminer avec la classe `Criteria`. On utilisera alors la méthode `getBestProvider(Criteria criteria, boolean enabledOnly)` afin d'obtenir le nom du fournisseur qui se rapproche le plus des critères demandés ou `getProviders(Criteria criteria, boolean enabledOnly)` qui retourne une liste de noms.

Pour obtenir la dernère localisation connue, on appellera la méthode `getLastKnownLocation()` qui retourne un objet de type `Location` dont on pourra extraire la latitude avec `double getLatitude()` et la longitude avec `double getLongitude()`.

La classe Location est ene classe de données représentant un emplacement géographique. L'emplacement peut comprendre une latitude, une longitude, un horodatage et d'autres informations telles que l'altitude, la vitesse, ...

Exemple :

```
...
LocationManager locationManager = null;
private String fournisseur;

private void initialiserLocalisation()
{
    if(locationManager == null)
    {
        locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
        Criteria criteres = new Criteria();

        // la précision  : (ACCURACY_FINE pour une haute précision ou ACCURACY_COARSE pour une moins bonne pré
    cision)
        criteres.setAccuracy(Criteria.ACCURACY_FINE);

        // l'altitude
        criteres.setAltitudeRequired(true);

        // la direction
        criteres.setBearingRequired(true);

        // la vitesse
```

 © 2017 tv <tvaira@free.fr>

```
        criteres.setSpeedRequired(true);

        // la consommation d'énergie demandée
        criteres.setCostAllowed(true);
        criteres.setPowerRequirement(Criteria.POWER_HIGH);

        fournisseur = locationManager.getBestProvider(criteres, true);
        Log.d("GPS", "fournisseur : " + fournisseur);
    }

    if(fournisseur != null)
    {
        // dernière position connue
        Location localisation = locationManager.getLastKnownLocation(fournisseur);
        Log.d("GPS", "localisation : " + localisation.toString());
        String coordonnees = String.format("Latitude : %f - Longitude : %f\n", localisation.getLatitude(),
    localisation.getLongitude());
        Log.d("GPS", "coordonnees : " + coordonnees);
        String autres = String.format("Vitesse : %f - Altitude : %f - Cap : %f\n", localisation.getSpeed(),
    localisation.getAltitude(), localisation.getBearing());
        Log.d("GPS", autres);
        //String timestamp = String.format("Timestamp : %d\n", localisation.getTime());
        //Log.d("GPS", "timestamp : " + timestamp);
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
        Date date = new Date(localisation.getTime());
        Log.d("GPS", sdf.format(date));
    }
}
...
```

L'interface LocationListener

Il est possible de créer une classe "écouteur" pour ce service en implémentant l'interface LocationListener qui reçoit les mises à jour de localisation par la méthode `onLocationChanged(Location location)`.

La méthode `onStatusChanged()` est appelée lorsqu'il y aura un changement d'état sur le fournisseur de localisation. Les méthodes `onProviderEnabled(String provider)` et `onProviderDisabled(String provider)` seront appelées quand le fournisseur de localisation est activé ou désactivé.

Exemple :

```
import android.location.*;

class MyLocationListener implements LocationListener
{
    public void onLocationChanged(Location localisation)
    {
        Log.d("GPS", "localisation : " + localisation.toString());
        String coordonnees = String.format("Latitude : %f - Longitude : %f\n", localisation.getLatitude(),
    localisation.getLongitude());
        Log.d("GPS", "coordonnees : " + coordonnees);
    }

    @Override
    public void onStatusChanged(String fournisseur, int status, Bundle extras)
    {
        Toast.makeText(, fournisseur + " état : " + status, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onProviderEnabled(String fournisseur)
    {
        Toast.makeText(, fournisseur + " activé !", Toast.LENGTH_SHORT).show();
    }

    @Override
```

 © 2017 tv <tvaira@free.fr>

```
    public void onProviderDisabled(String fournisseur)
    {
        Toast.makeText(, fournisseur + " désactivé !", Toast.LENGTH_SHORT).show();
    }
}
```

On a vu que pour obtenir la dernière position connue du terminal mobile, il fallait appeler `getLastKnownLocation(String provider)`.

Mais la dernière position connue n'est pas forcément la position actuelle ! Il faudra configurer la mise à jour automatique de la position en fonction de critères de temps et/ou de distance.

On pourra pour cela utiliser la méthode `requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener)` avec :

- `provider` le fournisseur de position ;
- `minTime` la période entre deux mises à jour en millisecondes (attention à la batterie) ;
- `minDistance` la période entre deux mises à jour en mètres ;
- `listener` est l'écouteur qui sera lancé dès que le fournisseur sera activé.

Remarque : attention à vos choix de `minTime` et de `minDistance` car ils vont influer sur la consommation d'énergie.

Exemple :

```
...
LocationManager locationManager = null;
private String fournisseur;
MyLocationListener myLocationListener; // écouteur

private void initialiserLocalisation()
{
    ...
    if(fournisseur != null)
    {
        // dernière position connue
        Location localisation = locationManager.getLastKnownLocation(fournisseur);
        myLocationListener = new MyLocationListener();

        if(localisation != null)
        {
            // on notifie la localisation
            myLocationListener.onLocationChanged(localisation);
        }

        // on configure la mise à jour automatique : au moins 10 mètres et 15 secondes
        locationManager.requestLocationUpdates(fournisseur, 15000, 10, myLocationListener);
    }
}
```

Lorsque le service est suspendu, il est indispensable de supprimer toutes les mises à jour de localisation avec la méthode `removeUpdates()` :

```
private void arreterLocalisation()
{
    if(locationManager != null)
    {
        locationManager.removeUpdates(myLocationListener);
        myLocationListener = null;
    }
}
```

La classe Geocoder

La classe Geocoder permet de gérer le géocodage et le géocodage inverse. Le géocodage est le processus de transformation d'une adresse postale (ou d'une autre description) d'un lieu en coordonnées (latitude, longitude) et le géocodage inverse est le processus de transformation d'une coordonnée (latitude, longitude) en une adresse (partielle). Les détails dans une description d'emplacement géocodée inversée peut varier.

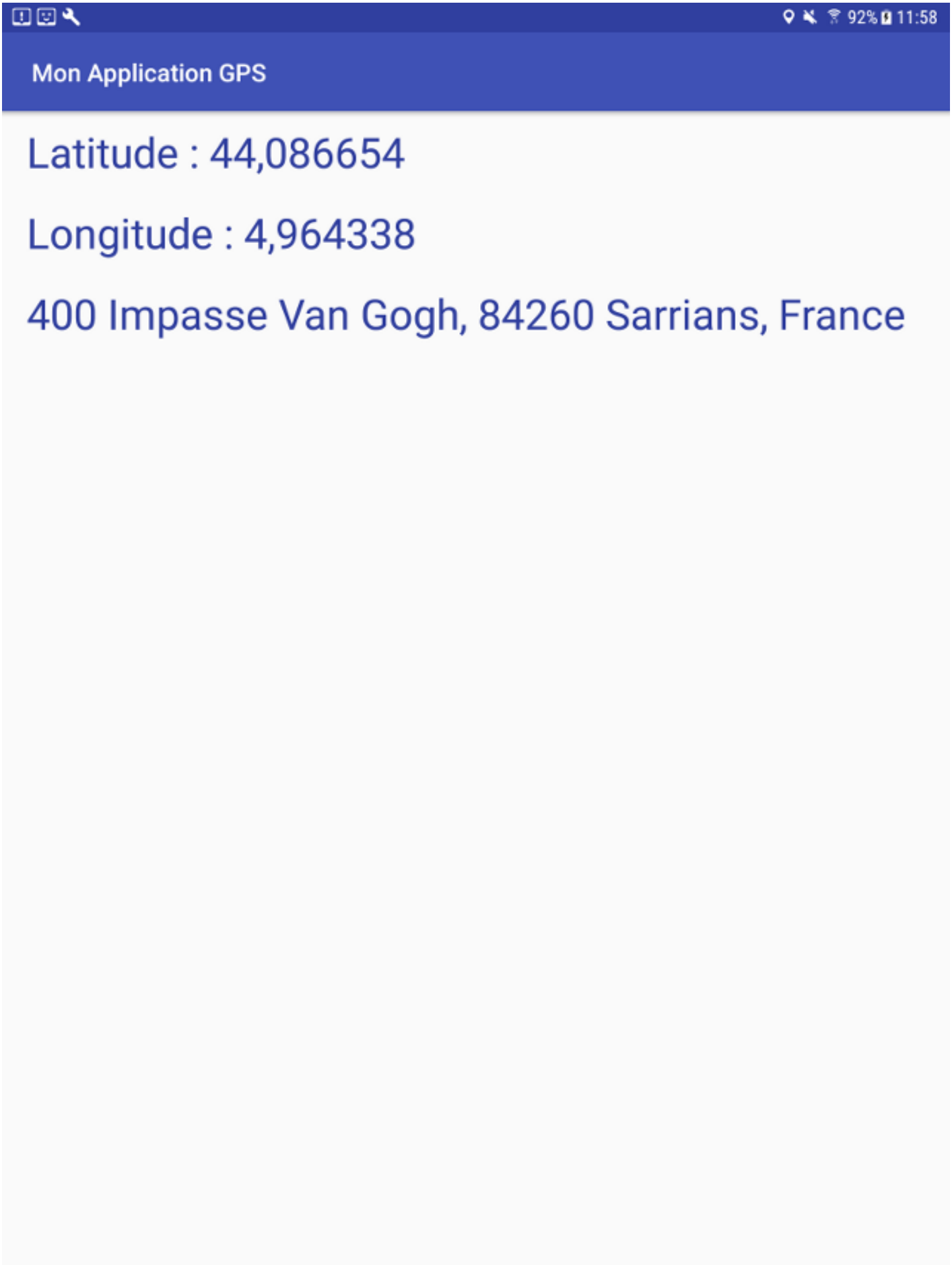
 © 2017 tv <tvaira@free.fr>

Exemple :

```
Geocoder geocoder = new Geocoder(this, Locale.getDefault());

List<Address> adresses = null;
try
{
    adresses = geocoder.getFromLocation(localisation.getLatitude(), localisation.getLongitude(), 1);
}
catch (IOException ioException)
{
    Log.e("GPS", "erreur", ioException);
}
catch (IllegalArgumentException illegalArgumentException)
{
    Log.e("GPS", "erreur " + coordonnees, illegalArgumentException);
}

if (adresses == null || adresses.size()  == 0)
{
    Log.e("GPS", "erreur aucune adresse !");
}
else
{
    Address adresse = adresses.get(0);
    ArrayList<String> addressFragments = new ArrayList<String>();

    for(int i = 0; i <= adresse.getMaxAddressLineIndex(); i++)
    {
        addressFragments.add(adresse.getAddressLine(i));
    }
    Log.d("GPS", TextUtils.join(System.getProperty("line.separator"), addressFragments));
}
```

 © 2017 tv <tvaira@free.fr>

Exemple



 © 2017 tv <tvaira@free.fr>

```
package com.example.tv.myapplicationgps;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.Criteria;
import android.location.LocationProvider;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;

public class MyActivity extends AppCompatActivity
{
    LocationManager locationManager = null;
    private String fournisseur;
    private TextView latitude;
    private TextView longitude;
    private TextView Adresse;

    Geocoder geocoder = new Geocoder(this, Locale.getDefault());

    LocationListener ecouteurGPS = new LocationListener() {
        @Override
        public void onLocationChanged(Location localisation)
        {
            Toast.makeText(getApplicationContext(), fournisseur + " localisation", Toast.LENGTH_SHORT).show();

            Log.d("GPS", "localisation : " + localisation.toString());
            String coordonnees = String.format("Latitude : %f - Longitude : %f\n", localisation.getLatitude(),
    localisation.getLongitude());
            Log.d("GPS", coordonnees);
            String autres = String.format("Vitesse : %f - Altitude : %f - Cap : %f\n", localisation.getSpeed(),
    localisation.getAltitude(), localisation.getBearing());
            Log.d("GPS", autres);
            //String timestamp = String.format("Timestamp : %d\n", localisation.getTime());
            //Log.d("GPS", "timestamp : " + timestamp);
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
            Date date = new Date(localisation.getTime());
            Log.d("GPS", sdf.format(date));

            String strLatitude = String.format("Latitude : %f", localisation.getLatitude());
            String strLongitude = String.format("Longitude : %f", localisation.getLongitude());
            latitude.setText(strLatitude);
            longitude.setText(strLongitude);

            List<Address> adresses = null;
            try
            {
                adresses = geocoder.getFromLocation(localisation.getLatitude(), localisation.getLongitude(), 1);
```

```
            }
            catch (IOException ioException)
            {
                Log.e("GPS", "erreur", ioException);
            } catch (IllegalArgumentException illegalArgumentException)
            {
                Log.e("GPS", "erreur " + coordonnees, illegalArgumentException);
            }

            if (adresses == null || adresses.size()  == 0)
            {
                Log.e("GPS", "erreur aucune adresse !");
            } else {
                Address adresse = adresses.get(0);
                ArrayList<String> addressFragments = new ArrayList<String>();

                String strAdresse = adresse.getAddressLine(0) + ", " + adresse.getLocality();
                Log.d("GPS", "adresse : " + strAdresse);

                for(int i = 0; i <= adresse.getMaxAddressLineIndex(); i++)
                {
                    addressFragments.add(adresse.getAddressLine(i));
                }
                Log.d("GPS", TextUtils.join(System.getProperty("line.separator"), addressFragments));
                Adresse.setText(TextUtils.join(System.getProperty("line.separator"), addressFragments));
            }
        }

        @Override
        public void onProviderDisabled(String fournisseur)
        {
            Toast.makeText(getApplicationContext(), fournisseur + " désactivé !", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onProviderEnabled(String fournisseur)
        {
            Toast.makeText(getApplicationContext(), fournisseur + " activé !", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onStatusChanged(String fournisseur, int status, Bundle extras)
        {
            switch(status)
            {
                case LocationProvider.AVAILABLE:
                    Toast.makeText(getApplicationContext(), fournisseur + " état disponible", Toast.LENGTH_SHORT
    ).show();
                    break;
                case LocationProvider.OUT_OF_SERVICE:
                    Toast.makeText(getApplicationContext(), fournisseur + " état indisponible", Toast.
    LENGTH_SHORT).show();
                    break;
                case LocationProvider.TEMPORARILY_UNAVAILABLE:
                    Toast.makeText(getApplicationContext(), fournisseur + " état temporairement indisponible",
    Toast.LENGTH_SHORT).show();
                    break;
                default:
                    Toast.makeText(getApplicationContext(), fournisseur + " état : " + status, Toast.
    LENGTH_SHORT).show();
            }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState)
```

```
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.commande);

    latitude = (TextView) findViewById(R.id.textViewLatitude);
    longitude = (TextView) findViewById(R.id.textViewLongitude);
    Adresse = (TextView) findViewById(R.id.textViewAdresse);

    Log.d("GPS", "onCreate");

    initialiserLocalisation();
}

@Override
protected void onDestroy()
{
    super.onDestroy();
    arreterLocalisation();
}

private void initialiserLocalisation()
{
    if (locationManager == null)
    {
        locationManager = (LocationManager) getApplicationContext().getSystemService(Context.
LOCATION_SERVICE);
        Criteria criteres = new Criteria();

        // la précision  : (ACCURACY_FINE pour une haute précision ou ACCURACY_COARSE pour une moins bonne
précision)
        criteres.setAccuracy(Criteria.ACCURACY_FINE);

        // l'altitude
        criteres.setAltitudeRequired(true);

        // la direction
        criteres.setBearingRequired(true);

        // la vitesse
        criteres.setSpeedRequired(true);

        // la consommation d'énergie demandée
        criteres.setCostAllowed(true);
        //criteres.setPowerRequirement(Criteria.POWER_HIGH);
        criteres.setPowerRequirement(Criteria.POWER_MEDIUM);

        fournisseur = locationManager.getBestProvider(criteres, true);
        Log.d("GPS", "fournisseur : " + fournisseur);
    }

    if (fournisseur != null)
    {
        // dernière position connue
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this, Manifest.permission.
ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
        {
            Log.d("GPS", "no permissions !");
            return;
        }

        Location localisation = locationManager.getLastKnownLocation(fournisseur);
        if(localisation != null)
        {
            // on notifie la localisation
            ecouteurGPS.onLocationChanged(localisation);
```

 © 2017 tv <tvaira@free.fr>

```
            }

            // on configure la mise à jour automatique : au moins 10 mètres et 15 secondes
            locationManager.requestLocationUpdates(fournisseur, 15000, 10, ecouteurGPS);
        }
    }

    private void arreterLocalisation()
    {
        if(locationManager != null)
        {
            locationManager.removeUpdates(ecouteurGPS);
            ecouteurGPS = null;
        }
    }
}
```

Liens

- Guide de développement Android
- Données de géolocalisation (Android API)
- La localisation et les cartes (OpenClassRooms)

 © 2017 tv <tvaira@free.fr>