GÉOLOCALISATION ET MAP

L'une des caractéristiques uniques des applications mobiles est la connaissance de l'emplacement. Les utilisateurs mobiles emmènent leurs appareils partout avec eux. En ajoutant la reconnaissance de la position à votre application, vous offrez une expérience plus contextuelle. Les API de localisation disponibles dans les services Google Play facilitent l'ajout de la reconnaissance de la localisation à votre application avec un suivi de localisation automatisé, une géo-repérage et une reconnaissance d'activité.

1.Obtenir le dernier emplacement connu

À l'aide des API de localisation des services Google Play, votre application peut demander le dernier emplacement connu du périphérique de l'utilisateur. Dans la plupart des cas, vous êtes intéressé par l'emplacement actuel de l'utilisateur, qui est généralement équivalent au dernier emplacement connu du périphérique.

En particulier, utilisez le fournisseur d'emplacement fusionné pour extraire le dernier emplacement connu du périphérique. Le fournisseur d'emplacement fusionné est l'une des API d'emplacement dans les services Google Play.

2.Spécifier les autorisations

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.location.sample.basiclocationsample" >
...
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
...
</manifest>
```

3.Créer le service de localisation Client

Première méthode :

Ensuite, on va faire appel à un nouveau service système pour accéder à ces fonctionnalités : LocationService, que l'on récupère de cette manière :

Fichier gradle :

```
dependencies {
...
implementation 'com.google.android.gms:play-services-location:16.0.0'
}
```

Java

Pour obtenir la dernière position connue de l'appareil, utilisez : `getLastLocation()`

```
FusedLocationProviderClient mClient=
LocationServices.getFusedLocationProviderClient(this.getApplicationContext()
);

mClient.getLastLocation()
        .addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can
be null.
                if (location != null) {
                    // Logic to handle location object
                }
            }
        });
```

Kotlin :

```
private lateinit var fusedLocationClient: FusedLocationProviderClient

    fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
}
```

```
fusedLocationClient.lastLocation
        .addOnSuccessListener { location : Location? ->
                if (location != null) {
                    // Logic to handle location object
                }
            // Got last known location. In some rare situations this can be
null.
        }
```

Seconde méthode

Java :

```
1.  LocationManager manager=( LocationManager ) this.getApplicationContext()
2.                                  .getSystemService(LOCATION_SERVICE);
3.
4.  Location maposition=manager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
5.  if(maposition==null)
6.              maposition=manager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER) ;
7.
8.  double  lng=maposition.getLongitude() ;
9.  double  lat=maposition.getLatitude() ;
```

Kotlin :

```
lateinit var locationService : LocationManager

/*******************/
locationService= getSystemService(Context.LOCATION_SERVICE) as LocationManager
```

Pour obtenir la dernière position connue de l'appareil, utilisez : `Location getLastKnownLocation(String provider)`

```
try {
     var gpslocation = locationmanager.getLastKnownLocation(
LocationManager.GPS_PROVIDER)

     var networklocation = locationmanager.getLastKnownLocation(
LocationManager.NETWORK_PROVIDER)

 }
catch(e:SecurityException)
{
    Toast.makeText(this,"Permission non accordé",Toast.LENGTH_LONG).show()
}
```

La dernière position connue n'est pas forcément la position actuelle de l'appareil. En effet, il faut demander à mettre à jour la position pour que celle-ci soit renouvelée dans le fournisseur. Si vous voulez faire en sorte que le fournisseur se mette à jour automatiquement à une certaine période ou tous les *x* mètres, on peut utiliser la méthode requestLocationUpdates

4.Recevoir des mises à jour d'emplacement

Première méthode :

Java :

```
1.  LocationCallback mcall = new LocationCallback(){
2.      @Override
3.      public void onLocationResult(LocationResult locationResult) {
4.          Toast.makeText(MainActivity.this,
5.                  locationResult.getLastLocation()+"///",
6.                  Toast.LENGTH_SHORT)
7.                  .show();
8.      }
9.  };
10. long i = 10;
11. float d = 10;
12. LocationRequest request = LocationRequest.create()
13.         .setInterval(i)
14.         .setSmallestDisplacement(d);
15.
16. mClient.requestLocationUpdates(request, mcall, null);
```

Kotlin :

```
class MyLocationCallBack : LocationCallback()
        {
            override fun onLocationResult(p0: LocationResult?) {

                if(p0!=null)
                {
                    var maposition= p0.lastLocation
                    var lng=maposition.longitude
                    var lat=maposition.latitude
                }

            }
        }
        var mcall=MyLocationCallBack()
        var mClient=LocationServices.getFusedLocationProviderClient(this)
        var request = LocationRequest.create()
                    .setInterval(1)
                    .setSmallestDisplacement(10F);

        mClient.requestLocationUpdates(request,mcall,null)
```

Seconde méthode :

On peut utiliser la méthode `void 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. Il faut mettre une valeur supérieure à 0, sinon le fournisseur ne sera pas mis à jour périodiquement. D'ailleurs, ne mettez pas de valeur en dessous de 60 000 ms pour préserver la batterie.
- `minDistance` la période entre deux mises à jour en mètres. Tout comme pour `minTime`, il faut mettre une valeur supérieure à 0, sinon ce critère ne sera pas pris en compte. De plus, on privilégie quand même `minTime` parce qu'il consomme moins de batterie.
- `listener` est l'écouteur qui sera lancé dès que le fournisseur sera activé.

Ainsi, il faut que vous utilisiez l'interface `LocationListener`, dont la méthode `void onLocationChanged(Location location)` sera déclenchée à chaque mise à jour.

Java :

```java
LocationListener myAction= new LocationListener(){

            @Override
            public void onLocationChanged(Location location) {
                double lat=location.getLatitude();
                double lng=location.getLongitude();
            }

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

            }

            @Override
            public void onProviderEnabled(String provider) {

            }

            @Override
            public void onProviderDisabled(String provider) {

            }
        };
        manager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                1,10,myAction);
```

Kotlin :

```kotlin
inner class MyLocationListener : android.location.LocationListener
{
    override fun onLocationChanged(location: Location?) {
        TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
    }
    override fun onStatusChanged(provider: String?, status: Int, extras: Bundle?)
{
        TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
    }
    override fun onProviderEnabled(provider: String?) {
        TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
    }
    override fun onProviderDisabled(provider: String?) {
        TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
    }
}
try {
    locationmanager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
        60000,100.toFloat(),  MyLocationListener())
}
    catch(e:SecurityException)
    {
        Toast.makeText(this,"Permission non accordé",Toast.LENGTH_LONG).show()
    }
```