Android: Services système

Année 2017-2018

Assane SECK
Ingénieur-Informaticien

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

SENSOR MANAGER

- Utilisé pour accéder et gérer le hardware senseur
- Exemple pour obtenir l'accéléromètre

```
mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

TYPES DE SENSEUR

- Sensor.TYPE_ACCELEROMETER
- Sensor.TYPE_GYROSCOPE
- Sensor.TYPE_LIGHT
- Sensor.TYPE_MAGNETIC_FIELD
- Sensor.TYPE_ORIENTATION
- Sensor.TYPE_PRESSURE
- Sensor.TYPE_PROXIMITY
- Sensor.TYPE_TEMPERATURE
- ...

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

OBTENIR LES SENSEURS

- Il peut y avoir plusieurs implémentations d'un senseur particulier
- Le senseur par défaut peut être trouvé en utilisant la fonction getDefaultSensor

OBTENIR LES SENSEURS : EXEMPLE

• Obtenir le senseur par défaut pour le gyroscope

```
Sensor defaultGyroscope=sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

• Obtenir tous les types de senseurs de pression

```
List<Sensor> pressureSensors =sensorManager.getSensorList(Sensor.TYPE_PRESSURE);
```

• Obtenir tous les senseurs

```
List<Sensor> allSensors =sensorManager.getSensorList(Sensor.TYPE_ALL);
```

UTILISER LES SENSEURS

- Les senseurs peuvent être implémentés en utilisant un SensorEventListener
- Utilisez la fonction onSensorChanged pour observer les valeurs du Sensoret onAccuracyChanged pour réagir aux changments dans la précision d'un Senseur

```
final SensorEventListener mySensorEventListener=new SensorEventListener()
{
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}
@Override
public void onSensorChanged(SensorEvent event) {

}
};
}
```

FONCTION onSensorChanged

- sensor : l'objet Sensor d'où provient l'event
- accuracy : la précision du senseur au moment où l'événement a eu lieu
- values : une array de float qui contient les nouvelles valeurs détectées

http://developer.android.com/intl/fr/reference/android/hardware/SensorEvent.html#values

- timestamp : le moment en nanosecondes auquel l'événement a eu lieu

FONCTION ONACCURACYCHANGED

- sensor : le senseur où la précision a changé

- précision
    - SensorManager.SENSOR_STATUS_ACCURACY_LOW
    - SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM
    - SensorManager.SENSOR_STATUS_ACCURACY_HIGH
    - SensorManager.SENSOR_STATUS_UNRELIABLE

S'ENREGISTRER POUR DES SENSOR EVENT

- Peut être fait en faisant appel à registerListener sur le SensorManager

```
Sensor sensor =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
sensorManager.registerListener(mySensorEventListener, sensor,
SensorManager.SENSOR_DELAY_NORMAL);
```

Fréquence de mise à jour

- SensorManager.SENSOR_DELAY_FASTEST
- SensorManager.SENSOR_DELAY_GAME
- SensorManager.SENSOR_DELAY_NORMAL
- SensorManager.SENSOR_DELAY_UI

ANNULER L'ENREGISTREMENT AUX SENSOR EVENTS

- Peut être fait en utilisant unregisterListener sur le SensorManager

```
sensorManager.unregisterListener(mySensorEventListener);
```

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

VIBRATION DU DEVICE

- La vibration peut être faite en utilisant les notifications, mais quoi si nous souhaitons vibrer de manière indépendante?
- Ajoutez la permission VIBRATE dans votre AndroidManifest.xml

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

- Obtenez le Vibrator Service, via getSystemService

```
String vibratorService = Context.VIBRATOR_SERVICE;
Vibrator vibrator = (Vibrator)getSystemService(vibratorService);
```

- Vibrez

```
 long[] pattern = {1000, 2000, 4000, 8000, 16000 };
vibrator.vibrate(pattern, 0); // Execute vibration pattern.
vibrator.vibrate(1000); // Vibrate for 1 second.
```

- Utilisez cancel() pour annuler la vibration.

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

TELEPHONY MANAGER : INITIER DES APPELS

- Meilleur pratique : lancer un dialer
- En utilisant un Intent
  - Action : Intent.ACTION_DIAL
  - Uri : Uri.parse("tel:123456")
- Exemple

```
Intent intent = new Intent(Intent.ACTION_DIAL,Uri.parse("tel:1234567"));

startActivity(intent);
```

- Afin de faire un appel directement, utilisez ACTION_CALL au lieu de ACTION_DIAL
- N'oubliez pas la permission CALL_PHONE dans votre manifest!

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

TELEPHONY MANAGER : ACTION_CALL

```
{
    int checkPermission = ContextCompat.checkSelfPermission(MapsActivity.this,
Manifest.permission.CALL_PHONE);
    if (checkPermission != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(
                MapsActivity.this,
                new String[]{Manifest.permission.CALL_PHONE},
                0);


    }
    Intent intent =
            new Intent(Intent.ACTION_CALL, Uri.parse("tel:000 00 00"));
    startActivity(intent);

    return false;
}
```

TELEPHONY MANAGER : SMS

```
{
    int checkPermission = ContextCompat.checkSelfPermission(MapsActivity.this,
Manifest.permission.SEND_SMS);
    if (checkPermission != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(
                MapsActivity.this,
                new String[]{Manifest.permission. SEND_SMS},
                0);


    }
    Intent intent =
            new Intent(Intent.ACTION_CALL, Uri.parse("tel:000 00 00"));
    startActivity(intent);

    return false;
}
```

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

VIBRATION DU DEVICE

- Vous pouvez accéder l'API de la téléphonie via le Telephony Manager
- Accessible via la fonction getSystemService()

```
String srvcName =Context.TELEPHONY_SERVICE;
TelephonyManager telephonyManager =
(TelephonyManager)getSystemService(srvcName);
```

LIRE LES DÉTAILS DU TÉLÉPHONE

- Vous pouvez obtenir le type du téléphone (GSM, CDMA), ID unique (IMEI or MEID), version software et numéro.
- Nécessite la permission READ_PHONE_STATE dans votre manifest
- Exemple

```
int phoneType = telephonyManager.getPhoneType();
switch
(phoneType)
{
case
(TelephonyManager.PHONE_TYPE_CDMA): break;
case (TelephonyManager.PHONE_TYPE_GSM): break;
case (TelephonyManager.PHONE_TYPE_NONE): break;
default: break;
}
```

```
These require READ_PHONE_STATE uses-permission
Read the IMEI for GSM or MEID for CDMA
String deveiceId=telephonyManager.getDeviceId();
Read the software version on the phone (note – not the SDK Version )
String softwareVersion=telephonyManager.getDeviceSoftwareVersion();
Get the phone's number
String phoneNumber=telephonyManager.getLine1Number();
```

LIRE L'ÉTAT DE LA CONNEXION ET TRANSFERT DE DONNÉES

- Utilisez les fonctions getDataState() et getDataActivity()
- Exemple

```
int dataActivity = telephonyManager.getDataActivity();
int dataState = telephonyManager.getDataState();

switch (dataActivity) {
case TelephonyManager.DATA_ACTIVITY_IN: break;
case TelephonyManager.DATA_ACTIVITY_OUT: break;
case TelephonyManager.DATA_ACTIVITY_INOUT: break;
case TelephonyManager.DATA_ACTIVITY_NONE: break;
}
switch (dataState) {
case TelephonyManager.DATA_CONNECTED: break;
case TelephonyManager.DATA_CONNECTING: break;
case TelephonyManager.DATA_DISCONNECTED: break;
case TelephonyManager.DATA_SUSPENDED: break;
}
```

LIRE LES DÉTAILS DU RÉSEAU

- Util pour lire le code pays et réseau, ISO code du pays, type de réseau auquel vous êtes connecté.

```
Get connected network country code
String networkCountry = telephonyManager.getNetworkCountryIso();

Get the connected network operator ID (MCC +MNC)
String networkOperatorId = telephonyManager.getNetworkOperator();

Get the connected network operator name
String networkName=telephonyManager.getNetworkOperatorName();

Get the type of network you are  connected to
int networkType= telephonyManager.getNetworkType();
```

LIRE L'ÉTAT DE LA CARTE SIM

- Util pour lire le code pays et réseau, ISO code du pays, type de réseau auquel
- vous êtes connecté.

```
TelephonyManager
telMgr=(TelephonyManager)getSystemService(TELEPHONY_SERVICE);
int simState=telMgr.getSimState();
switch (simState) {
case TelephonyManager.SIM_STATE_ABSENT: break;
case TelephonyManager.SIM_STATE_NETWORK_LOCKED: break;
case TelephonyManager.SIM_STATE_PIN_REQUIRED: break;
case TelephonyManager.SIM_STATE_PUK_REQUIRED: break;
case TelephonyManager.SIM_STATE_UNKNOWN: break;
case TelephonyManager.SIM_STATE_READY:
{
String simCountry= telMgr.getSimCountryIso(); //Get the operator code of the active SIM (MCC+MNC)
String simOperatorCode=telMgr.getSimOperator();
String simOperatorName=telMgr.getSimOperatorName();//Requires READ_PHONE_STATE uses-permission Get the SIM's serial number
String simSerial =telMgr.getSimSerialNumber();
}
}
```

OBSERVER DES CHANGEMENTS DANS L'ÉTAT DU TÉLÉPHONE

- Nécessite la permission READ_PHONE_STATE
- Les changements dans l'état du téléphone peuvent être observés en utilisan la classe PhoneStateListener
    - Implémentez et écouter pour les event:
    - Etat des appels
    - Changement d'antenne
    - Voice mail
    - Redirection d'appels

PHONESTATELISTENER

- Exemple d'implémentation:

```
PhoneStateListener phoneStateListener = new PhoneStateListener() {
public void onCallForwardingIndicatorChanged(boolean cfi) {
}
public void onCallStateChanged(int state, String incomingNumber) {
}
public void onCellLocationChanged(CellLocation location) {
}
public void onDataActivity(int direction) {
}
public void onDataConnectionStateChanged(int state) {
}
public void onMessageWaitingIndicatorChanged(boolean mwi) {
}
public void onServiceStateChanged(ServiceState serviceState) {
}
public void onSignalStrengthChanged(int asu) {
}
};
```

PHONESTATELISTENER

- Attacher un listener:

```
telMgr.listen(phoneStateListener,
PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR
| PhoneStateListener.LISTEN_CALL_STATE
| PhoneStateListener.LISTEN_CELL_LOCATION
| PhoneStateListener.LISTEN_DATA_ACTIVITY
| PhoneStateListener.LISTEN_DATA_CONNECTION_STATE
| PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR
| PhoneStateListener.LISTEN_SERVICE_STATE
| PhoneStateListener.LISTEN_SIGNAL_STRENGTH);
```

- Se désenregistrer

```
telMgr.listen(phoneStateListener, PhoneStateListener.LISTEN_NONE);
```

Sommaire

SERVICES

Sensor manager
Types de senseurs
Utiliser des senseurs
Service vibration
Initier un appel
Etat de la téléphonie
SMS manager

SENDING AN SMS THROUGH INTENTS

- Utilise le client SMS natif
- Exemple

```
Intent smsIntent = new Intent(Intent.ACTION_SENDTO,
Uri.parse("sms:55512345"));
smsIntent.putExtra("sms_body", "Press send to send me");
startActivity(smsIntent);
```

• Vous pouvez ajouter un Intent.EXTRA_STREAM avec l'Uri vers la ressource que vous souhaitez attacher, accompagnée de son MIME-type, afin d'envoyer un MMS.

ENVOYER UN SMS MANUELLEMENT

- L'envoi d'SMS dans Android est géré par le SmsManager
- Vous pouvez obtenir une référence vers celui-ci en faisant appel à la fonction statique:

```
SmsManager smsManager = SmsManager.getDefault();
```

• Vous aurez également besoin de la permission SEND_SMS

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

ENVOYER UN SMS

- Utilisez la fonction sendTextMessage() depuis le SMS Manager en donnant l'adresse (numéro de téléphone) du récipient et le message.

```
SmsManager smsManager = SmsManager.getDefault();
String sendTo = "5551234";
String myMessage = "Android supports programmatic SMS messaging!";
smsManager.sendTextMessage(sendTo, null, myMessage, null, null);
```

- Le deuxième paramètre est le SMSC, par défaut null

- Les deux derniers paramètres sont des intents afin d'assurer le suivi par rapport à la transmission et l'envoi de votre message.

SUIVI DE TRANSMISSION ET LIVRAISON D'SMS

• Afin d'assurer le suivi de la transmission et la livraison, implémentez et enregistrez des BroadcastReceivers qui écoutent les messages que vous spécifiez dans les PendingIntent que vous donnez à la fonction ssendTextMessage

- Paramètre sentIntent : lorsque le message est envoyé
  - Activity.RESULT_OK
  - SmsManager.RESULT_ERROR_GENERIC_FAILURE
  - SmsManager.RESULT_ERROR_RADIO_OFF
  - SmsManager.RESULT_ERROR_NULL_PDU

• deliveryIntent n'est seulement lancé lorsque le récipient recoit l'SMS

ASSURER LE SUIVI SMS (1/3)

- Définir les Intents

```
String SENT_SMS_ACTION = "SENT_SMS_ACTION";
String DELIVERED_SMS_ACTION = "DELIVERED_SMS_ACTION";
//Create the sentIntent parameter
Intent sentIntent=new Intent(SENT_SMS_ACTION);
PendingIntent sentPI=PendingIntent.getBroadcast(getApplicationContext(), 0, sentIntent, 0);
//Create the deliveryIntent parameter
Intent deliveryIntent=new Intent(SENT_SMS_ACTION);
PendingIntent deliveryPI=PendingIntent.getBroadcast(getApplicationContext(), 0,
deliveryIntent, 0);
```

ASSURER LE SUIVI SMS (2/3)

- Enregistrez l'Activity pour broadcast action SENT_SMS_ACTION

```
registerReceiver(new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
switch (getResultCode()) {
case Activity.RESULT_OK:
//Send success action
break;
case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
//generic failure
break;
case SmsManager.RESULT_ERROR_RADIO_OFF:
//Radio off failure action
break;
case SmsManager.RESULT_ERROR_NULL_PDU:
//null PDU failure actions
break;
}
}
},
new IntentFilter(SENT_SMS_ACTION));
}
```

ASSURER LE SUIVI SMS (3/3)

- Enregistrez votre Activity pour la DELIVERED_SMS_ACTION

```
registerReceiver(new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
}
},
new IntentFilter(DELIVERED_SMS_ACTION));
}
```

- Envoyez le message

```
smsManager.sendTextMessage(sendTo, null, myMessage, sentPI, deliverPI);
```

RÉCEPTION DE MESSAGES SMS

• Lorsqu'un nouvel SMS est reçu par le téléphone, un nouveau broadcast event est lancé avec l'action android.provider.Telephony.SMS_RECEIVED.

• Afin de pouvoir recevoir ces intent, vous devez spécifier la permission RECEIVE_SMS dans votre manifeste.

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

• Le broadcast message SMS contient les détails de l'SMS entrant.

EXTRAIRE LES INFORMATIONS D'UN SMS ENTRANT

- Exemple

```
if (bundle != null) {
final Object[] pdusObj = (Object[]) bundle.get("pdus");
SmsMessage []messages=new SmsMessage[pdusObj.length];
for (int i = 0; i < pdusObj.length; i++) {
 messages[i]=SmsMessage.createFromPdu((byte[])pdusObj[i]);
}
}
```

SERVEUR ECHO SMS

```
public class IncomingSms extends BroadcastReceiver {
private static final String queryString = "@echo";
private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
public void onReceive(Context context, Intent intent) {
if (intent.getAction().equals(SMS_RECEIVED)) {
SmsManager sms = SmsManager.getDefault();
Bundle bundle = intent.getExtras();
if (bundle != null) {
final Object[] pdusObj = (Object[]) bundle.get("pdus");
SmsMessage[] messages = new SmsMessage[pdusObj.length];
for (int i = 0; i < pdusObj.length; i++) {
messages[i] = SmsMessage.createFromPdu((byte[]) pdusObj[i]);

for (SmsMessage message : messages) {
String msg = message.getMessageBody();
String to = message.getOriginatingAddress();

if (msg.toLowerCase().startsWith(queryString)) {
String out = msg.substring(queryString.length());
sms.sendTextMessage(to, null, out, null, null);
}}}}}}}
```

FIN