

Contents

1. Receiving
 1. Catch the new SMS message
 2. Make an activity
 3. Others
2. Sending
 1. Define SmsManager
 2. Divide SMS message (Optional)
 3. Process Dialog (Optional)
 4. Actual sending
3. Searching
 1. Example
 1. open a cursor
 2. Get all SMSs actual message and the number
 2. Other
 1. before you write
 2. open a cursor
 3. getting people's real name and their phone number
 4. mapping someone's number to their phone number
4. License

SMS Messaging by Yunseok Jang

Updated on: 03.29.2009

SMS has 3 big parts, which are receiving, sending, searching(or seeing). So I'll handle all of them one by one.

Receiving

This part is for doing something when you got a new SMS message. For doing this, you need to write some code at your AndroidManifest.xml.

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

And you need to define this part also (I'll write receiving part at "SMSReceiver.class" file)

```
<receiver android:name="SMSReceiver" android:enabled="true">
  <intent-filter>
```

```
<action
android:name="android.provider.Telephony.SMS_RECEIVED" />
</intent-filter>
</receiver>
```

As you can see, receiving action is not an activity, so some of the part is different to activity(for example, making an intent...). After you write these part at your AndroidManifest.xml, you need to make a class for handing receive part. General structure of receiving part is following.

Toggle line numbers

```
1 public class SMSReceiver extends BroadcastReceiver {
2     public void onReceive(Context context, Intent intent)
3     {
4         if (intent.getAction().equals(ACTION)) {
5             // body
6         }
7     }
```

Catch the new SMS message

If you want to get the information of the new SMS (sender/body/etc), you need to write this part at body part.

Toggle line numbers

```
1 Bundle bundle = intent.getExtras();
2
3 if (bundle != null) {
4     Object[] pduObj = (Object[]) bundle.get("pdu");
5     SmsMessage[] messages = new
SmsMessage[pduObj.length];
6
7     // getting SMS information from Pdu.
8     for (int i = 0; i < pduObj.length; i++) {
9         messages[i] =
SmsMessage.createFromPdu((byte[]) pduObj[i]);
10    }
11
12    for (SmsMessage currentMessage : messages) {
13        //
currentMessage.getDisplayOriginatingAddress() has sender's phone
number
14        // currentMessage.getDisplayMessageBody()
has the actual message
15    }
```

```
16 }
```

As you can see, you can get sender's address and actual message.

Make an activity

If you want to make an activity, you need to add a flag at your intent. (Reason : receiver is different from activity as I said before!!)

Toggle line numbers

```
1 Intent i = new Intent(context, 'put your class name to  
here'.class);  
2 i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
3 context.startActivity(i);
```

As general intent do, you need to define your activity at AndroidManifest.xml also.
For example,

```
<activity android:name="your class name"></activity>
```

Others

You can also make a toast, make a macro that works for specific SMS, etc.

Sending

When you want to send SMS message, you need to write uses-permission at your AndroidManifest.xml

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

After doing that, you can use SMS sending and handling functions.

Define SmsManager

The first thing that you need to do for sending SMS is defining an SmsManager.

Toggle line numbers

```
1 SmsManager smsMgr = SmsManager.getDefault();
```

Divide SMS message (Optional)

After you define your own SmsManager, you can divide your SMS message by 160 bytes. (Optional, if your SMS always less than 160 bytes, you can pass this part)

Toggle line numbers

```
1 ArrayList<String> messages = smsMgr.divideMessage(smsText);
```

smsText (String) has the SMS that we want to send. You can get total number of messages by using messages.size().

Process Dialog (Optional)

You can also make a ProgressDialog like that.(Optional)

Toggle line numbers

```
1 dlg = new ProgressDialog(this);
2 dlg.setIndeterminate(true);
3 dlg.setCancelable(false);
4 dlg.setMessage("Sending " + messages.size() + " messages.");
5
6 SMSSender.this.handler.postDelayed(new Runnable() {
7     public void run() {
8         dlg.show();
9     }
10 }, 2000);
```

Actual sending

Toggle line numbers

```
1 public static final String
SMS_ADDRESS_PARAM="SMS_ADDRESS_PARAM";
2 public static final String
SMS_DELIVERY_MSG_PARAM="SMS_DELIVERY_MSG_PARAM";
3 public static final String
SMS_SENT_ACTION="com.tilab.msn.SMS_SENT";
4 for (String address : addresses) {
```

```
5         ArrayList<PendingIntent> listOfIntents = new
ArrayList<PendingIntent>();
6         for (int i=0; i < messages.size(); i++){
7             Intent sentIntent = new
Intent (SMS_SENT_ACTION);
8             sentIntent.putExtra (SMS_ADDRESS_PARAM,
address);
9             sentIntent.putExtra (SMS_DELIVERY_MSG_PARAM,
(messages.size() > 1)? "Part " + i + " of SMS " : "SMS ");
10            PendingIntent pi =
PendingIntent.getBroadcast (this, 0, sentIntent,
PendingIntent.FLAG_CANCEL_CURRENT);
11            listOfIntents.add(pi);
12        }
13
14        smsMgr.sendMultipartTextMessage (address, null,
messages, listOfIntents, null);
15 }
```

addresses is `ArrayList<String>` that has addresses for sending SMS. If you want to send a SMS to only one person, you can delete first for loop.

You can also use 'sendTextMessage' rather than 'sendMultipartTextMessage', but actually `sendMultipartTextMessage` also support sending 1 SMS message, so I recommend to use `sendMultipartTextMessage`. (you can see <http://developer.android.com/reference/android/telephony/gsm/SmsManager.html> for more information)

Note : address can have '+1' or '-'

Searching

Before the 1.0 release of the SDK ([see thread](#)), one could search SMS easily by using 'android.provider.telephony.sms'. However, this provider was removed after the update, so we need to use some other ways; open an cursor and find it.

But you still need to write this part at your `AndroidManifest.xml` file.

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

The simplest example is to open a `Cursor` in this way:

Toggle line numbers

```
1 Cursor curSms = managedQuery (Uri.parse ("content://sms"),
```

```
    null, null, null, null);
```

You can open each SMS message part by opening different ways.

Toggle line numbers

```
1 Inbox = "content://sms/inbox"
2 Failed = "content://sms/failed"
3 Queued = "content://sms/queued"
4 Sent = "content://sms/sent"
5 Draft = "content://sms/draft"
6 Outbox = "content://sms/outbox"
7 Undelivered = "content://sms/undelivered"
8 All = "content://sms/all"
9 Conversations = "content://sms/conversations"
```

And these Cursors has the following columns:

Toggle line numbers

```
1 addressCol= mCurSms.getColumnIndex("address");
2 personCol= mCurSms.getColumnIndex("person");
3 dateCol = mCurSms.getColumnIndex("date");
4 protocolCol= mCurSms.getColumnIndex("protocol");
5 readCol = mCurSms.getColumnIndex("read");
6 statusCol = mCurSms.getColumnIndex("status");
7 typeCol = mCurSms.getColumnIndex("type");
8 subjectCol = mCurSms.getColumnIndex("subject");
9 bodyCol = mCurSms.getColumnIndex("body");
```

Note: "person" is the Contact ID of the corresponding person, not the actual Contact Name. You have to open a Cursor with People.CONTENT_URI and get the value in "display_name" column.

Some values are null, so make sure you check them.

Example

open a cursor

Toggle line numbers

```
1 Uri uri = Uri.parse("content://sms/inbox");
2 Cursor c= getContentResolver().query(uri, null, null
, null, null);
3 startManagingCursor(c);
```

Get all SMSs actual message and the number

Toggle line numbers

```
1 body = new String[c.getCount()];
2 number = new String[c.getCount()];
3
4 if(c.moveToFirst()){
5     for(int i=0;i<c.getCount();i++){
6         body[i]=
c.getString(c.getColumnIndexOrThrow("body")).toString();
7
8         number[i]=c.getString(c.getColumnIndexOrThrow("address")).toString
();
9         c.moveToNext();
10 }
11 c.close();
```

Other

Sometimes, we want to know users name rather than phone number. So I write how to get actual name from contacts.

before you write

You need to write permission at your AndroidManifest.xml file.

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

open a cursor

Toggle line numbers

```
1 String key, value;
2 String columns[] = new String[] { People.NAME, People.NUMBER
};
3 Uri mContacts = People.CONTENT_URI;
4 Cursor cur = managedQuery(mContacts, columns, null, null,
null);
```

getting people's real name and their phone number

Toggle line numbers

```
1 ActualSender = new Hashtable<String, String> ();
2 if (cur.moveToFirst ()) {
3     do {
4         value =
cur.getString (cur.getColumnIndex (People.NAME));
5         key =
cur.getString (cur.getColumnIndex (People.NUMBER));
6         if (key!=null)
7             ActualSender.put (key, value);
8     } while (cur.moveToNext ());
9 }
```

Note : Some contacts have no number(for example : place that user saved from Google map).


key has '-' also(ex key = "781-XXX-XXXX", you'd better remove the dash)

mapping someone's number to their phone number

You can use `ActualSender.containsKey(phone_number)`, which returns true if it has this phone number, otherwise return false.

Note : sometimes, phone_number have +1 but not the contact list.

License

This work is licensed under a  Creative Commons Attribution-NonCommercial-Share Alike 3.0 United States License.



CategoryTutorial

FrontPage/Tutorials/SMS Messaging (last edited 2009-03-30 04:36:38 by MarkChang)