Android Programing

Gavrilut Dragos

SMSManager

- An object that can be used to send SMS messages programatically
- Any message send using this object will not appear in the outbox
- The following permission needs to be added to the manifest:

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

```
public void SendSMS(String number,String text)
{
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(number, null, text, null, null);
}
```

Receiving a SMS message

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.RECEIVE_SMS" />
- The following code should be added to the manifest to register a class that will be listening to the messages that are send:

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

- Create the class MySMSReceiver to intercept SMS messages

Receiving a SMS message

```
public class SMSReceiver extends BroadcastReceiver
{
    public void onReceive(Context context, Intent intent) {
        if ( intent.getExtras() != null )
        {
            Object[] smsextras = (Object[]) intent.getExtras().get( "pdus" );
            for (int tr = 0; tr < smsextras.length; tr++ )
            {
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[])smsextras[tr]);

                String text = smsMessage.getMessageBody().toString();
                String number = smsMessage.getOriginatingAddress();

                 // record text and number
             }
         }
    }
}
```

- Use **abortBroadcast()** this message from beeing send to other receivers.

Listing all SMS messages

```
public List<MySMSMessage> getAllSms() {
        List<MySMSMessage> listSms = new ArrayList<MySMSMessage>();
        MySMSMessage obj = new MySMSMessage();
        Uri message = Uri.parse("content://sms/");
        ContentResolver cr = this.getContentResolver();

        Cursor c = cr.query(message, null, null, null, null);
        this.~~startManagingCursor(c);~~
        int totalSMSMessages = c.getCount();

        if (c.moveToFirst()) {
            for (int i = 0; i < totalSMSMessages; i++) {

                obj = new MySMSMessage();
                obj.Id = c.getString(c.getColumnIndexOrThrow("_id"));
                obj.Address = c.getString(c.getColumnIndexOrThrow("address"));
                obj.Body = c.getString(c.getColumnIndexOrThrow("body"));
                obj.ReadState = c.getString(c.getColumnIndex("read")).equals("1");
                obj.Time = c.getString(c.getColumnIndexOrThrow("date"));
                obj.Inbox = c.getString(c.getColumnIndexOrThrow("type")).contains("1");
                listSms.add(obj);
                c.moveToNext();
            }
        }
        c.close();

        return listSms;
    }
```

```
public class MySMSMessage
{
            public String Id;
            public String Address;
            public String Body;
            public boolean ReadState;
            public String Time;
            public boolean Inbox;
}
```

Delete a SMS message

- Two methods:
  - Directly on your receiver class by using abordBroadcast() function
  - Programatically – but you need to know the SMS message ID:

```
public boolean DeleteSMS(String smsId) {
    try {
        this.getContentResolver().delete(Uri.parse("content://sms/" + smsId), null, null);
        return true;
    }
    catch (Exception ex) {
        return false;
    }
}
```

- The following permission needs to be added to the manifest:

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

Get all contacts list

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.READ_CONTACTS“/>

```
private void ReadAllContacts()
{
    ContentResolver cRes = getContentResolver();
        Cursor c = cRes.query(ContactsContract.Contacts.CONTENT_URI,null, null, null, null);
        if (c.getCount() > 0) {
            while (c.moveToNext()) {
                String id = c.getString(c.getColumnIndex(ContactsContract.Contacts._ID));
                String name = c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
                String tmp = c.getString(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
                int numberID = Integer.parseInt(tmp);
                if (numberID > 0)
                {
                    Cursor result = cRes.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = ?",
                                        new String[]{id}, null);
                    while (result.moveToNext())
                    {
                        String phoneNumber = result.getString(result.getColumnIndex(
                                                    ContactsContract.CommonDataKinds.Phone.NUMBER));
                       // use the phoneNumber
                    }
                    result.close();
                }
            }
        }
    }
```

Create a contact

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.WRITE_CONTACTS“/>

```
 private void CreateContact(String name, String phoneNumber) {
        ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
        Builder b;

        b = ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_URI);
        b = b.withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, "abc@gmail.com");
        b = b.withValue(ContactsContract.RawContacts.ACCOUNT_NAME, "...");
        ops.add(b.build());

        b = ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI);
        b = b.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0);
        b = b.withValue(ContactsContract.Data.MIMETYPE,
                        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
        b = b.withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,name);
        ops.add(b.build());

        b = ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI);
        b = b.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0);
        b = b.withValue(ContactsContract.Data.MIMETYPE,
                        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
        b = b.withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, phoneNumber);
        b = b.withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
                        ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);
        ops.add(b.build());

        try {
            getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
        } catch (Exception e) { … }
}
```

Delete a contact

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.WRITE_CONTACTS“/>

```
private void DeleteContact(String name)
{
        ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
        Builder b = ContentProviderOperation.newDelete(ContactsContract.RawContacts.CONTENT_URI);
        b = b.withSelection(ContactsContract.Data.DISPLAY_NAME + " = ? ", new String[] {name});
        ops.add(b.build());

        try
        {
            getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
        }
        catch (Exception e)
        {
            // ...
        }
}
```

Blocking a phone-call

- The following permission needs to be added to the manifest:
  <uses-permission android:name="android.permission.PHONE_STATE" />
  <uses-permission android:name="android.permission.NEW_OUTGOING_CALL" />
- The following code should be added to the manifest to register a class that will be listening to the calls that are made:

```
<receiver android:name="MyPhoneReceiver">
    <intent-filter android:priority="100">
        <action android:name="android.intent.action.PHONE_STATE"/>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
    </intent-filter>
</receiver>
```

- Create the class MyPhoneReceiver to intercept call messages

Receiving a phone call

```
public class ProcessCall extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        TelephonyManager telephony;
        telephony = (TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
        MyPhoneStateListener listener = new MyPhoneStateListener (context);
        telephony.listen(listener, PhoneStateListener.LISTEN_CALL_STATE);
    }
}

public class MyPhoneStateListener extends PhoneStateListener {
    Context context;
    public MyPhoneStateListener(Context context) {
        super();
        this.context = context;
    }

    @Override
    public void onCallStateChanged(int state, String callingNumber)
    {
        super.onCallStateChanged(state, callingNumber);
        if ((state == TelephonyManager.CALL_STATE_OFFHOOK) ||
            (state == TelephonyManager.CALL_STATE_RINGING))
            {
                 BlockTheCall();
            }
   }
}
```

Blocking a call - undocumented

```
private void BlockCall(String callingNumber)
{
    try
    {
        TelephonyManager tm;
        tm = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
        Class c = Class.forName(tm.getClass().getName());
        Method m = c.getDeclaredMethod("getITelephony");
        m.setAccessible(true);
        com.android.internal.telephony.ITelephony telephonyService = (ITelephony) m.invoke(tm);
        telephonyService = (ITelephony) m.invoke(tm);
        telephonyService.silenceRinger();
        telephonyService.endCall();
    }
    catch (Exception e) {

    }
 }
```

E nevoie de link static la clasa com.android.internal sau apel prin reflexie