宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

By 阳哥

Android 手机卫士-03

1.检查 SIM 卡是否更换（★★★）

原理：1、sim 插入一般会拔掉电源，会导致开启重启；2、侧面卡槽也会导致手机重启--重启通讯模块；因此我们只需要监听手机重启事件，当手机重启的时候检查当前 sim 卡的序列号是否跟已经绑定的手机 sim 卡序列号一致即可。

步骤：1、创建新包 com.itheima.mobileSafe.receiver，根据名称大家也推测到该包下主要存放 Receiver 类。

2、在该包下创建 BootCompleteReceiver 类，该类用户监听手机的启动事件。

3、在 AndroidManifest.xml 中注册 BootCompleteReceiver。

```
<receiver android:name="com.itheima.mobileSafe.receiver.BootCompleteReceiver" >
          <intent-filter>
              <action android:name="android.intent.action.BOOT_COMPLETED" />
          </intent-filter>
</receiver>
```

4、在 AndroidManifest.xml 中声明权限。在 Android 系统中监听系统开机事件是需要如下权限的：

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

5、编写 BootCompleteReceiver 类

```
public class BootCompleteReceiver extends BroadcastReceiver {

    private TelephonyManager tm;
    private SharedPreferences sp;
    @Override
    public void onReceive(Context context, Intent intent) {
        /*
         * 从 sp 中获取是否开启手机保护信息
         * 如果没有开启则直接返回
         */
```

```
        sp = context.getSharedPreferences("config", Context.MODE_PRIVATE);
        boolean proteccting = sp.getBoolean("protectting", false);
        if (!proteccting) {
            return ;
        }
        tm = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
        /*
         * 从 sp 中获取已经绑定的 sim 卡序列号
         * 如果没有绑定 sim 卡则返回
         */
        String sim = sp.getString("sim", "");
        if (TextUtils.isEmpty(sim)) {
            System.out.println("SIM 卡未绑定");
            Toast.makeText(context, "SIM 卡未绑定", Toast.LENGTH_SHORT).show();
            return;
        }else{
            /*
             * 通过 TelephonyManager 对象获取当前 sim 卡的序列号
             * 如果当前的 sim 卡和 sp 中的 sim 卡序列号一致则可以断定 SIM 卡没有更换
             * 如果两次的 SIM 卡序列号不一致，则断定 SIM 已经更换，这时可以根据用户的设置给安
             * 全号码发送短信，通知 SIM 卡已经更换
             */
            String serialNumber = tm.getSimSerialNumber();
            if (sim.equals(serialNumber)) {
                System.out.println("sim 卡依然没变");
                Toast.makeText(context, "SIM 卡未更换", Toast.LENGTH_SHORT).show();
            }else {
                //发送短信给安全号码
                System.out.println("sim 卡已经变更");
                Toast.makeText(context, "SIM 卡已经更换", Toast.LENGTH_SHORT).show();
                SmsManager.getDefault().sendTextMessage(sp.getString("saveNumber",
"", null, "sim 卡已经更改，from antaojin", null,null);
                System.out.println("已经给安全号码发送短信"+sp.getString("saveNumber",
""));
            }
        }
    }
}
```

6、关闭模拟器，重新启动则可以启动该功能，不过由于模拟器中没有 sim 卡，因此我们可以放在我们的真机中进行测试。

2.读取手机联系人（★★★★）

在上节中当 SIM 卡信息改变后程序会给安全号码发送一条通知短信。在设置向导页面的第三个页面(如下图所示)，让用户指定一个安全号码，该安全号码既可以是手动输入也可以是通过系统联系人选取。

步骤：1、编写设置安全号码界面的布局文件 activity_setup3.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:background="#8866ff00"
        android:gravity="center"
        android:text="3 设置安全号码"
        android:textColor="#000000"
        android:textSize="20sp" />
    <TextView
        style="@style/text_content_style"
       android:text="SIM 卡变更后：\n 报警短信会发给安全号码" />
```

```
    <EditText
        android:id="@+id/et_save_number"
        android:hint="请输入安全号码"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:onClick="selectContact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="选择联系人"
        android:background="@drawable/button_select_contact"
        />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal" >
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_online" />
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />
    </LinearLayout>
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/phone" />
```

```
        <Button
            android:id="@+id/button1"
            style="@style/button_next" />
        <Button
            android:id="@+id/button2"
            style="@style/button_previous" />
    </RelativeLayout>
</LinearLayout>
```

2、编写获取联系人界面的布局文件 activity_select_contact.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:background="#8866ff00"
        android:gravity="center"
        android:text="选择联系人"
        android:textColor="#000000"
        android:textSize="20sp" />
    <ListView
        android:id="@+id/list_contact"
        android:verticalSpacing="10dp"
        android:layout_marginTop="10dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>
</LinearLayout>
```

3、编写 Setup3Activity 类，实现核心逻辑功能

```
/**
 * 设置安全号码
 * @author wzy Jan 4, 2015
 *
 */
```

```
public class Setup3Activity extends BaseSetupActivity {
    private EditText et_save_number;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_setup3);
        /*
         * 从 sp 中获取安全号码，如果已经设置安全号码则进行数据的回显
         */
        et_save_number = (EditText) findViewById(R.id.et_save_number);
        String number = sp.getString("saveNumber", null);
        if (!TextUtils.isEmpty(number)) {
            et_save_number.setText(number);
        }
    }

    public void next(View view) {
        showNext();
    }
    public void previous(View view) {
        showPrevious();
    }
    @Override
    public void showNext() {
        /*
         * 进入下一步设置前必须让用户先设置安全号码
         * 如果没有设置安全号码则不允许进入下一步
         */
        String number = et_save_number.getText().toString();
        if (TextUtils.isEmpty(number)) {
            Toast.makeText(this, "请设置安全号码", Toast.LENGTH_SHORT).show();
            return ;
        }
        //进入下一步设置
        Intent intent = new Intent(this, Setup4Activity.class);
        startActivity(intent);
        finish();
        overridePendingTransition(R.anim.tran_in, R.anim.tran_out);
    }
    @Override
    public void showPrevious() {
        Intent intent = new Intent(this, Setup2Activity.class);
        startActivity(intent);
```

```
        finish();
        overridePendingTransition(R.anim.tran_pre_in, R.anim.tran_pre_out);
    }
    /*
     * 打开系统联系人界面，当从系统联系人选择号码号后重新返回当前界面
     */
    public void selectContact(View view){
        Intent intent = new Intent(this, SelectContactActivity.class);
        //调用 startActivityForResult 方法，并覆写 onActivityResult 方法，用于接收第二个
Activity 返回的数据
        startActivityForResult(intent, 100);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode==RESULT_OK) {
            /*
             * 接收第二个 Activity 返回来的安全号码
             * 将该安全号码保存到 sp 中
             * 将该安全号码显示在当前界面的编辑框中可以供用户再次编辑
             */
            String number = data.getStringExtra("number");
            et_save_number.setText(number);
            Editor editor = sp.edit();
            editor.putString("saveNumber", number);
            editor.commit();
        }
    }
}
```

4、编写 SelectContactActivity 类，在该类中让用户选择安全号码，并把安全号码发送给 Setup3Activity 类，这里用到了 Activity 之间数据的传递知识。

```
public class SelectContactActivity extends Activity {
    private ListView list_contact;
    Handler handler = new Handler() {
        @SuppressWarnings("unchecked")
        @Override
        public void handleMessage(Message msg) {
            if (msg.what == RESULT_OK) {
                // 从 Message 中拿到发送的消息
```

```
            final List<Map<String, String>> data = (List<Map<String, String>>)
msg.obj;
            /*
             * 给 ListView 设置一个 SimpleAdapter
             * SimpleAdapter 第一个参数是 Context 对象
             * 第二个参数是 List<? extends Map<String, ?>>集合
             * 第三个参数是 int 类型，其实就是对应着 ListView 显示的每一个条目的布局文件
id
             * 第四个参数是 String[]类型，里面对应着 map 中的 key
             * 第五个参数是 int[]类型，里面对应着 map 中 key 应该在每个条目布局文件中的显
示控件的 id
             */
            list_contact.setAdapter(new SimpleAdapter(SelectContactActivity.this,
data, R.layout.contact_item_view, new String[] { "name", "phone" }, new int[]
{ R.id.tv_name, R.id.tv_phone }));
            /*
             * 给 ListView 对象设置点击事件
             */
            list_contact.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                    //获取点击的位置，拿到对应的数据
                    Map<String, String> map = data.get(position);
                    //获取电话号码
                    String number = map.get("phone");
                    //去掉电话号码中的“-”
                    number = number.replace("-", "");
                    Intent intent = new Intent();
                    //将号码绑定在 intent 中
                    intent.putExtra("number", number);
                    setResult(RESULT_OK, intent);
                    //销毁当前 Activity
                    SelectContactActivity.this.finish();
                }
            });
        }
    }
};
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_select_contact);
        list_contact = (ListView) findViewById(R.id.list_contact);
        getContacts();
    }
    // 获取联系人,因为获取联系人可能是耗时操作，因此需要在子线程中进行
    private void getContacts() {
        // 将联系人信息保存在 Map 中，然后将该 map 存在 List 集合中
        final List<Map<String, String>> list = new ArrayList<Map<String, String>>();
        // 获取系统的 ContentResolver 对象
        final ContentResolver contentResolver = getContentResolver();
        new Thread(new Runnable() {
            @Override
            public void run() {
                Uri uri = Uri.parse("content://com.android.contacts/raw_contacts");//
id 联系人被删除后为 contact_id 为 null，deleted 为 1
                Uri datauri = Uri.parse("content://com.android.contacts/data");
                Cursor cursor = contentResolver.query(uri, new String[] { "contact_id",
"deleted" }, null, null, null);
                while (cursor.moveToNext()) {
                    Map<String, String> map = new HashMap<String, String>();
                    // 获取联系人 id
                    String id = cursor.getString(0);
                    if (id != null) {
                        // 根据联系人的 id 在 data 表中获取联系人的数据以及数据类型
                        Cursor cursordata = contentResolver.query(datauri, new String[]
{ "mimetype", "data1" }, "raw_contact_id=?", new String[] { id }, null);
                        // 定义一个标识记录联系人数据是否为空
                        boolean flag = false;
                        while (cursordata.moveToNext()) {
                            String mimetype = cursordata.getString(0);
                            String data1 = cursordata.getString(1);
                            // 如果 mimeType 类型为电话则将该数据保存到 map 中
                            if ("vnd.android.cursor.item/phone_v2".equals(mimetype)) {
                                map.put("phone", data1);
                                flag = true;
                            } else if ("vnd.android.cursor.item/name".equals(mimetype)) {
                                // 如果是姓名则保存在 map 中
                                map.put("name", data1);
                            }
                        }
                        cursordata.close();
                        // 如果获取到了当前联系人的电话（姓名可以无）则将 map 添加到 list 中
```

```
                    if (flag) {
                        list.add(map);
                        flag = false;
                    }
                }
            }
            cursor.close();
            // 创建一个新的 Message 对象
            Message msg = new Message();
            msg.what = RESULT_OK;
            // 将数据添加给 Message
            msg.obj = list;
            // 发送消息给主线程
            handler.sendMessage(msg);
        }
    }).start();
  }
}
```

注意：在上面的代码中我们的 ListView 的每一个条目用到了一个布局文件，该布局文件清单如下：

contact_item_view.xml 布局清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/tv_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="姓名"
        android:textColor="#ff0000"
        android:textSize="18sp" />
    <TextView
        android:id="@+id/tv_phone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="联系电话"
        android:textColor="#000000"
        android:textSize="18sp" /></LinearLayout>
```

注意：在上面的代码中我们读取了用户联系人，因此需要添加对应的权限。

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

3.手机防盗的原理（★）

手机防盗主要是通过用户更换 SIM 卡时程序自动给我们的安全号码发送短信、发送位置信息等功能获取手机当前的位置、SIM 卡等信息。同时我们也可以通过其他手机给我们的手机发送短信指令，根据不同的指令安全卫士会做一些锁屏、删除数据、恢复出厂设置等操作以保护我们手机数据的安全。

4.短信指令的广播接收者（★★★）

创建一个短信指令广播接收者，用于接收通过短信发送来的指令，然后根据指令的内容执行不同的操作。这也是远程锁屏、远程清空数据的核心业务逻辑。

步骤：1、在 com.itheima.mobileSafe.receiver 包下，我们创建 SMSReceiver 类继承 BroadcastReceiver 类。

2、在 AndroidManifest.xml 中注册该接收者

```
<receiver android:name="com.itheima.mobileSafe.receiver.SMSReceiver" >
            <intent-filter android:priority="1000" >
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
 </receiver>
```

上面配置信息中我们给了一个 android:priority="1000"属性，该属性值最大 1000 最小-1000，这里我们把优先级设为最大，则最先接收到短信（在新版本中无法终止该广播，在老版本中可以终止该广播）。

3、添加接收短信的权限

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

4、编写 SMSReceiver 类，在该类中实现核心方法

```
public class SMSReceiver extends BroadcastReceiver {
    private SharedPreferences sp;
    //设备策略管理器
    private DevicePolicyManager dpm;
    @Override
    public void onReceive(Context context, Intent intent) {
        sp = context.getSharedPreferences("config", Context.MODE_PRIVATE);
        //获取系统策略管理器对象
        dpm = (DevicePolicyManager)
context.getSystemService(Context.DEVICE_POLICY_SERVICE);
        String saveNumber = sp.getString("saveNumber", null);
        //通过 intent 获取发送短信的内容
        Object[] objs = (Object[]) intent.getExtras().get("pdus");
        for(Object obj : objs){
            //获取短信对象
            SmsMessage pdu = SmsMessage.createFromPdu((byte[])obj);
            //获取短信内容
            String msg = pdu.getMessageBody();
            //获取短信发信人号码
            String address = pdu.getOriginatingAddress();
            //如果是来自安全号码的短信则拦截
            if (address.contains(saveNumber)) {
                Toast.makeText(context, "收到安全号码发来的短信：body "+msg, 0).show();
                //如果短信内容是 location 则发送位置信息给安全号码
                if ("location".equals(msg)) {
                    //开启位置服务功能，位置服务获取位置信息后将最新位置放在 sp 中，这样我们
就可以从 sp 中获取到最新的位置信息
                    Intent intentGPS = new Intent(context, GPSservice.class);
                    context.startService(intentGPS);
                    String location = sp.getString("location", null);
                    String dest = sp.getString("saveNumber", "");
                    //获取不到位置信息时发送的消息
                    if (TextUtils.isEmpty(location)) {
                        SmsManager.getDefault().sendTextMessage(dest, null, "getting
location... from wzy lost", null, null);
                    }else {
                        //获取到位置信息后发送位置信息给安全号码
                        SmsManager.getDefault().sendTextMessage(dest, null,
location+"from wzy lost", null, null);
                    }
                    //终止该短信广播
```

```
                abortBroadcast();
            }else if ("alarm".equals(msg)) {
                //如果发送的是 alarm 内容，咋播放音乐
                playAlarm(context);
                abortBroadcast();
            }else if ("wipedata".equals(msg)) {
                System.out.println("删除数据");
                wipeData(context);
                abortBroadcast();
            }else if (msg!=null&&msg.startsWith("lockscreen")) {
                System.out.println("锁屏");
                int indexOf = msg.indexOf("_");
                if (indexOf>0&&msg.length()>indexOf+1) {
                    String pwd = msg.substring(indexOf+1, msg.trim().length());
                    dpm.resetPassword(pwd, 0);
                }
                dpm.lockNow();
                abortBroadcast();
            }
        }else {
            System.out.println("收到短信：from "+address+" body "+msg);
            Toast.makeText(context, "收到短信：from "+address+" body "+msg,
0).show();
        }
    }
}
//清除数据
private void wipeData(Context context) {
    dpm.wipeData(0);
}
//播放报警音乐
private void playAlarm(Context context) {
    MediaPlayer player = MediaPlayer.create(context, R.raw.ylzs);
    try {
        player.prepare();
    } catch (IllegalStateException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    player.setVolume(1.0f, 1.0f);
    player.setLooping(true);
```

```
        player.start();
    }
}
```

注意：由于上面的 SMSReceiver 类需要在 AndroidManifest.xml 中注册，这里就省去注册步骤。

在上面代码中我们用到了清除数据、一键锁屏功能，这些功能。是我们本文档的重要知识点，我们会在本文档的第 6 节中详细说明。

5.获取手机位置（★★）

在上一节中我们通过短信指令可以将手机的位置坐标获取并发送给安全号码。那么这一节我们将重点讨论 Android 系统如何获取位置信息。

手机定位方式有多种，常见的有 GPS 定位、网络定位。

1. GPS 定位　　精确度高，仅适用于户外，严重消耗电量。如果手机内置 GPS 接受模块，即使手机处于信号盲区，依然可以获取位置信息。

2. NETWORK_PROVIDER　网络定位，室内室外都可以使用，响应速度快，耗电量少。

在我们的项目中，系统会根据当前条件（有无 GPS、网络等）自动选择最优定位方案。

在 com.itheima.mobileSafe.service 包（如果还没有则创建）下新建一个 GPSservice（注意：这里我们给类起的名字不太好了，可能会让大家误以为我们就是用 GPS 定位技术）类继承 Service 类。

然后在 AndroidManifest.xml 中注册该 Service。

GPSservice 类代码清单如下：

```
public class GPSservice extends Service {
    //位置管理器 定位的核心类
    private LocationManager lm;
    private SharedPreferences sp;
```

```
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        //获取系统位置服务对象
        lm = (LocationManager) getSystemService(LOCATION_SERVICE);
        sp = getSharedPreferences("config", MODE_PRIVATE);
        //创建位置条件筛选对象
        Criteria criteria = new Criteria();
        //设置精度类型，这里是最佳精度
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        //通过位置管理器获取最好的位置服务提供者，这里是由 Android 系统自动提供，返回字符串
代表当前最佳定位服务提供者的名称
        String bestProvider = lm.getBestProvider(criteria , true);
        /*
         * 给位置管理器注册位置监听
         * 第一个参数代表位置服务提供者名称
         * 第二个参数代表最小时间间隔 单位毫秒
         * 第三个参数代表最小距离间隔 单位米
         */
        lm.requestLocationUpdates(bestProvider, 0, 0, new LocationListener() {
            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
            }
            //位置服务器可以使用时调用的方法
            @Override
            public void onProviderEnabled(String provider) {
                Toast.makeText(getApplicationContext(), provider+"已经可以使用.",
1).show();
            }
            //位置服务器不可以使用时调用的方法
            @Override
            public void onProviderDisabled(String provider) {
                Toast.makeText(getApplicationContext(), provider+"不可以使用.",
1).show();
            }
            //位置改变时调用的方法
            @Override
            public void onLocationChanged(Location location) {
                //获取纬度信息
                double latitude = location.getLatitude();
```

```
                //获取精度信息
                double longitude = location.getLongitude();
                //获取定位提供者的名称
                String provider = location.getProvider();
                System.out.println(provider+"提供的位置为：经度："+longitude+"\n"+"维度："
"+latitude);
                Toast.makeText(getApplicationContext(), provider+"提供的位置为：经度："
"+longitude+"\n"+"维度："+latitude, 0).show();
                //将位置信息保存在 sp 中
                Editor editor = sp.edit();
                editor.putString("location", longitude+"_"+latitude);
                editor.commit();
            }
        });
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
    }
}
```

上面的代码需要使用带权限。

6.一键锁屏&清除手机数据（★★★★）

从 Android 2.2 开始，引入了支持企业应用的设备管理 API。这些 API 支持我们的应用对系统密码进行操作（设置密码、设置密码安全等级等）以及清除系统数据等。

步骤：

1、在工程的 res 目录下创建一个 xml 目录，在 xml 目录下新建一个 xml 文件，名字叫 device_admin.xml。

device_admin.xml 清单如下：

该文件里面的清单，代表着我们申请的用户策略列表。

```
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-policies>
    <limit-password />
    <watch-login />
    <reset-password />
    <force-lock />
    <wipe-data />
    <expire-password />
    <encrypted-storage />
    <disable-camera />
  </uses-policies>
</device-admin>
```

2、在 com.itheima.mobileSafe.receiver 包下定义一个 MyDeviceAdminReceiver 类继承 DeviceAdminReceiver 类。因为这里我们不需要监听设备管理器的任何动作，因此我们可以不覆写任何方法。

3、在 AndroidManifest.xml 中注册该 Receiver。

```
<receiver
            android:name="com.itheima.mobileSafe.receiver.MyDeviceAdminReceiver"
            android:description="@string/sample_device_admin_description"
            android:label="@string/sample_device_admin"
            android:permission="android.permission.BIND_DEVICE_ADMIN" >
            <meta-data
                android:name="android.app.device_admin"
                android:resource="@xml/device_admin" />
            <intent-filter>
                <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
            </intent-filter>
</receiver>
```

注意：如果我们没有用代码打开设备管理，那么需要我们的用户手动打开，在 System settings->Security->Device administrators 中把对应的设备勾选上才可以。但是这样的操作对于一般用户来说太麻烦了，因此我们需要用代码激活设备管理员权限。

4、开启设备管理员权限代码如下：

```
public void admin(View view){
      Intent intent = new Intent(DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
      ComponentName componentName = new ComponentName(this,
MyDeviceAdminReceiver.class);
      intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN, componentName);
       intent.putExtra(DevicePolicyManager.EXTRA_ADD_EXPLANATION,"获取系统管权限
");
       startActivityForResult(intent, 110);
   }
```

我们可以将上面的代码放在设置向导中。

至此，本文档完！

2015 年 1 月 5 日 星期一 21:55:31
北京市海淀区东馨园小区